

拒絶理由通知書

特許出願の番号	特願2001-205181
起案日	平成16年10月19日
特許庁審査官	川崎 優 8944 5P00
特許出願人代理人	志賀 正武(外 1名) 様
適用条文	第36条、第37条

この出願は、次の理由によって拒絶をすべきものである。これについて意見があれば、この通知書の発送の日から3か月以内に意見書を提出して下さい。

理 由

1. この出願は、下記の点で特許法第37条に規定する要件を満たしていない。

記

請求項1に記載される発明が解決しようとする課題は、エンコーダバッファ及びデコーダバッファならびにエンコーダ及びデコーダの状態を検出して、ホストから送られてきた命令の実行を制御することに対して、請求項4に記載される発明が解決しようとする課題は、ホストから送られてきた命令が完成された命令語であるかどうかを判断し、命令語を最終的に組み立てるための構成を提供することであると認められる。

よって、請求項1に記載される発明、請求項4に記載される発明は、それぞれの解決しようとする課題が同一でなく、特許法第37条第1号に規定する関係を有するとは認められない。

また、請求項1に記載される発明の主要部は、エンコーダバッファ、デコーダバッファ、エンコーダ、デコーダの状態を保持する作業状態レジスタ及び制御管理部であり、請求項4に記載される発明の主要部は、請求項4のステップ(a) - (c)、すなわち、

(a) 外部ホストシステムから入力されたデータが制御命令語であるのか伝送データであるのかを区分する段階と、

(b) 制御命令語ならば、完成された命令であるのかを確認して、続く命令語があれば、制御命令語を累積した後で前記(a)段階に進行し、続く命令語がなければ、命令識別レジスタのあらかじめ割当てられたビットに、実行されるべき命令語を記入する段階と、

(c) 前記命令識別レジスタのあらかじめ割当てられた他のビットに、エンコーダ及び/またはデコーダが行う命令があることを示す段階、

BEST AVAILABLE COPY

であると認められる。

よって、請求項1に記載される発明、請求項4に記載される発明は、それぞれの主要部が相違するから、特許法第37条第2号に規定する関係を有すると認められない。

さらに、各発明は、特許法第37条第3号、第4号、第5号に規定する関係のいずれを満たすものとも認められない。

この出願は特許法第37条の規定に違反しているので、請求項1-3, 6以外の請求項に係る発明については新規性、進歩性等の要件についての審査を行っていない。

2. この出願は、特許請求の範囲の記載が下記の点で、特許法第36条第4項、第6項第1号、第2号に規定する要件を満たしていない。

記

明細書の特許請求の範囲の請求項（以下、請求項）1には、「エンコーダ及びデコーダが行わねばならない命令語を記録する命令識別レジスタ」という記載がある。

ところで、「行わねばならない」という記載は技術的に見て意味不明である（「実行されるべき」などとする方がよい）。

また、当該箇所についての発明の詳細な説明の裏付けとして、段落0013には「次に、ICM110は・・・CIR130に該当命令語を記録する」とあるものの、同段落におけるc i r（CIRの所定ビット）の説明や、段落0020-0022におけるc i rの説明を見る限りでは、CIRはエンコーダ及びデコーダが実行すべき命令がどのような状態にあるかという情報を保持しているものの、命令語そのものを記録しているわけではない。

してみると、請求項1の発明は明細書に裏付けられていないということもできるし、請求項1の発明を実施可能に開示していないということもできる。

また、請求項3には「エンコーダ及びデコーダによって行われる1以上の命令」「行われる命令数」という記載があるが、これらも同様に技術的に見て意味不明である（「実行される」などとする方がよい）

拒絶の理由が新たに発見された場合には拒絶の理由が通知される。

先行技術文献調査結果の記録

・調査した分野 IPC第7版 G06F9/30, 9/38, 15/16-167, H04N7/24-68

DB名

・先行技術文献

- 1.特開平11-85969号公報
- 2.特開平10-135842号公報
- 3.特開平9-18871号公報
- 4.鈴木ほか, 高性能ビデオプロセッサコアアーキテクチャの開発(1)～アーキテクチャ概要～, 電子情報通信学会2000年総合大会講演論文集, 2000年 3月 7日, P.124

この先行技術文献調査結果の記録は、拒絶理由を構成するものではない。

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-018871

(43)Date of publication of application : 17.01.1997

(51)Int.Cl.

HO4N	7/24
GO6F	9/38
HO4N	1/41
HO4N	7/015
// HO3M	7/30

(21)Application number : 07-266747

(71)Applicant : DISCOVISION ASSOC

(22)Date of filing : 13.09.1995

(72)Inventor : ADRIAN PHILIP WISE
WILLIAM PHILIP ROBBINS
JONES ANTHONY MARK
CLAYDON ANTHONY PETER JOHN
MARTIN WILLIAM SOTHERAN

(30)Priority

Priority number : 94 9405914 Priority date : 24.03.1994 Priority country : GB
95 9504047 28.02.1995

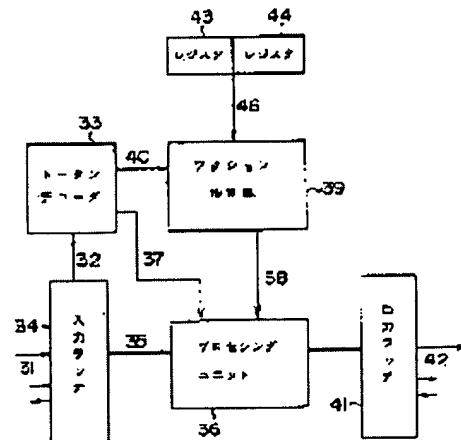
GB

(54) RECONSTRUCTABLE PROCESSING SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To provide reinforced flexibility to the improvement of the expansion method and device acting like decoding and/or expanding plural different coded input signals.

SOLUTION: A multiple standard video decoder is provided with plural processing stages 36 arranged as pipeline processing units interconnected by a 2-wire interface. A control token and a data token are sent via 2-wire interfaces 31, 42 to send both control signal and data in a token format. A token decoding circuit 33 is placed on a specific stage to recognize a specific tone as a control token specific to the specific stage and to send a control token not recognized along with a pipeline. A reconstruction processing circuit reconstructs the selected stage and processes recognized and discriminated data token in response to the recognized control token placed on the selected stage.



LEGAL STATUS

[Date of request for examination]

13.09.1995

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the withdrawal
examiner's decision of rejection or application
converted registration]

[Date of final disposal for application] 11.05.1999

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of
rejection]

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-18871

(43) 公開日 平成9年(1997)1月17日

(51) Int. Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
H04N 7/24			H04N 7/13	Z
G06F 9/38	310		G06F 9/38	E
H04N 1/41			H04N 1/41	B
7/015		9382-5K	H03M 7/30	Z
// H03M 7/30			H04N 7/00	A
審査請求 有 請求項の数 3 F D (全369頁)				

(21) 出願番号 特願平7-266747
 (62) 分割の表示 特願平7-90010の分割
 (22) 出願日 平成7年(1995)3月24日

(31) 優先権主張番号 9405914. 4
 (32) 優先日 1994年3月24日
 (33) 優先権主張国 イギリス (GB)

(31) 優先権主張番号 9504047. 3
 (32) 優先日 1995年2月28日
 (33) 優先権主張国 イギリス (GB)

(71) 出願人 591226829
 ディスコビジョン アソシエイツ
 アメリカ合衆国, カリフォルニア州 9271
 4, アーバイン, スウィート 200, メイン
 ・ストリート 2355

(72) 発明者 エイドリアン フィリップ ワイズ
 イギリス国, ビーエス16 1エヌエー, プ
 リストル, フレンチェイ, ウェストボーン
 コテージズ 10

(74) 代理人 弁理士 伊藤 嘉昭

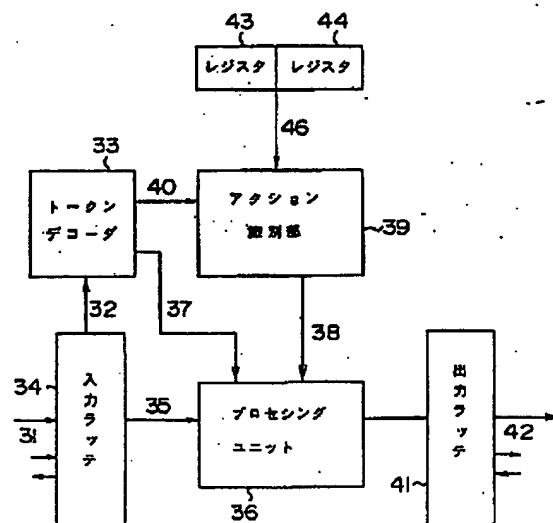
最終頁に続く

(54) 【発明の名称】 再構成可能なプロセッシングシステム

(57) 【要約】

【目的】 本発明は複数の異なる符号化入力信号を復号及び/又は伸長するように機能する伸長方法及び装置の改良に関する。

【構成】 多重規格ビデオ復号化装置は2配線インターフェースにより接続され、パイプライン処理装置として配置された複数の処理ステージ36を有する。制御トークンとデータトークンは制御とデータの両方をトークンフォーマットで送信するために2配線インターフェース31, 42を介して送られる。トークン復号化回路33は特定のトークンを特定ステージ固有の制御トークンとして認識するため、認識されなかった制御トークンをパイプラインに沿って送るためにこの特定のステージに置かれている。再構成処理回路は選択されたステージに置かれ認識された制御トークンに応答してその様なステージを再構成し認識された識別済みのデータトークンを扱う。



【特許請求の範囲】

【請求項1】 デジタル式のピクチャ情報処理システムにおいて、

複数の異なるピクチャ圧縮／伸長規格に基づきデータを処理するために前記システムを選択的に構成するための手段を具備することを特徴とするピクチャ情報処理システム。

【請求項2】 複数の処理ステージを有するシステムにおいて、

前記処理ステージ間における制御機能及び／又はデータ機能のための相互作用インターフェーストークンの形態の万能適応ユニットを具備し、

前記処理ステージは構成及び処理において強化された柔軟性が与えられることを特徴とするシステム。

【請求項3】 入力、出力、前記入力と前記出力との間の複数の処理ステージとを具備するシステムにおいて、前記処理ステージ間における制御機能及び／又はデータ機能のための万能適応ユニットを定義する相互作用メタモルフィックインターフェーストークンを具備し、前記処理ステージは種々のタスクの性能において強化された柔軟性が与えられることを特徴とするシステム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 この発明は、複数の異なる符号化入力信号を復号及び／又は伸長するように機能する伸長方法及び装置の改良に関する。以下に記載する実施例は説明的なものとして選択されたものであり、複数の符号映像の標準を符号化に関する。特に、この実施例はJ P E G、M P E G及びH. 2 6 1として知られる公知の標準の一つの復号化に関する。

【0002】 本発明のシリアルパイプライン処理方式は、独特且つ特殊な相互作用のインターフェーストークンを制御トークンとデータトークンの形態で、再構成可能パイプラインプロセッサとして位置付けされている複数の適応伸長回路等を実施するために使用される1個の2線バスから構成される。

【0003】

【従来の技術】 従来技術の一つに、米国特許第5, 2 1 6, 7 2 4号明細書に記載されているものがある。この装置は複数の計算モジュールから成り、好ましい実施例では、合計4個の計算モジュールが並列に接続されている。個々の計算モジュールは、処理装置と、デュアルポートメモリと、スクラッチパッドメモリと、アービトレーション機構で構成されている。第1のバスはこれらの計算モジュールとホストプロセッサを接続する。この装置は、ホストプロセッサと計算モジュールに第2のバスを介して接続された共用メモリを有する。

【0004】 米国特許第4, 7 8 5, 3 4 9明細書には、コンパクトディスク媒体に記録され従来のビデオフレーム速度で復号される、送信用の完全フォーマットを

開示している。圧縮中、フレームの領域は、各領域固有の最適なフィルコーディング方法を選択するために個々に分析される。領域復号化時間を推定し圧縮閾値を最適化する。領域の大きさと位置を示す領域記述コードはデータストリームの第1セグメントでグループ化される。領域の画素振幅を示す領域フィルコードはフィルコードの種類によりグループ化され、データストリームのその他のセグメントに置かれる。各データストリームセグメントはそれぞれの統計分布に応じて符号化された可変長を有し、データフレーム形成のためにフォーマット化される。フレームのバイト数は逆フレーム列分析により決定される補助データの追加により少なくなり、コンパクトディスクの再生中の一時停止を最少にするために選択される平均数を提供する。これによりコンパクトディスクの予測不可能なシークモード待機時間特性を回避する。デコーダは、データストリームのそれぞれのセグメントを別々に可変長符号するためのコードストリームの統計的情報に応答する可変長のデコーダを含む。領域位置データは領域記述データから得られ、フィルコードの種類（例えば相対値、絶対値、2項及びD P C M）を検出することにより選ばれる複数の領域特定デコーダに対して領域フィルコードと共に供給される。そして復号された領域画素は、後の表示用にビットマップ形式で格納される。

【0005】 米国特許第4, 9 2 2, 3 4 1明細書には、デジタルTV信号用の画像データをシーン・モデルを利用して縮小する方法が開示されている。ここでは、順次供給される画像信号は符号化され、時間 $t-1$ で既に符号化されているシーンからの前フレームは基準として画像格納部に存在し、フレームからフレームの情報増幅係数、移動係数及び適応的に得られたクアド

(4本) ツリー分割構造からなる。システムが初期化されると、均一な、所定の階調値又は規定の輝度値としてハーフ・トーン表現された画像が、送信機側のコードの画像格納部及び受信機側のデコーダの画像格納部に全ての画素（ピクセル）に対して同様に書き込まれる。コードの画像格納部及びデコーダの画像格納部の両方はそれぞれに対するフィードバックにより、コード及びデコーダ内の画像格納部の内容が可変サイズのブロックで読み出され、輝度値1より大きいかそれ未満の係数により増幅され、この画像格納部の別のアドレスに書き込み可能に動作する。それにより、可変サイズのブロックは公知のクアドツリー構造に於いて編成される。

【0006】 米国特許第5, 1 2 2, 8 7 5号明細書には、H D T V信号の符号化／復号化装置が開示されている。この装置は、圧縮されたビデオデータを表わす階層構造の符号化語C Wと、この符号化語C Wが表わすデータの種類を規定する、対応する符号化語Tを提供するための高品質ビデオソース信号に応答する圧縮回路を含む。符号化語C WとTに応答する優先選択回路は符号化

語 CW を高優先度及び低優先度符号化語列に分解し、それにより高位及び低位の優先符号化語列は、夫々画像再生に対して比較的重要度の高いそして低い圧縮ビデオデータに対応することになる。高優先度及び低優先度符号化語列に 대응する転送プロセッサは、それぞれ高優先度及び低優先度符号化語の高優先度及び低優先度転送ブロックを形成する。各転送ブロックはヘッダと、符号化語 CW とエラー検出チェックビットを含む。個々の転送ブロックは、追加のエラーチェックデータを供給する前方エラーチェック回路に供給する。その後、この高優先度及び低優先度データはモデムに供給され、対応する送信用の搬送波を直交振幅変調する。

【0007】米国特許第 5, 146, 325 号明細書には、奇数及び偶数フィールドのビデオ信号がフレーム内圧縮モードとフレーム間圧縮モードで順番に独立して圧縮され、次に送信用にインターリーブされた圧縮画像データを復号するビデオ復号システムが開示されている。奇数及び偶数フィールドは独立して復号される。有効な復号された奇数／偶数フィールドデータが無い期間では、偶数／奇数フィールドデータは利用不可の奇数／偶数フィールドデータに代用される。偶数／奇数フィールドのデータを独立して復号し、反対のフィールドのデータを利用不可のデータに代用することは、システム起動時及びチャンネル変化中の画像表示待機時間を減少するには有効に使用され得る。

【0008】米国特許第 5, 168, 356 号明細書には、符号化されたビデオデータを信号送信用の転送ブロックに分割するビデオ信号符号化システムが開示されている。送信されたデータが紛失又は壊れたときに受信機がデータストリームへの再進入ポイントを決定可能にするヘッダデータ提供のおかげで、その転送ブロックフォーマットは受信機側での信号回復を向上させる。再進入ポイントは、対応する転送ブロック内の符号化ビデオデータ中に挿入される二次的転送ヘッダを提供することにより最大になる。

【0009】米国特許第 5, 168, 375 号明細書には、デシメーション(decimation)、補間及び整形機能の内の一つ又はそれ以上の機能を提供するために 1 フィールド分の画像データサンプルを処理する方法が開示されている。これは J P E G 圧縮システムに利用されているアレキ変換プロセッサ等により実現される。データサンプルのブロックは、デシメーション及び補間処理の両方で離散偶数コサイン変換(D E C T)により変換され、その後、周波数項の数が変更される。デシメーションの場合は、周波数項の数は減少され、この後、逆変換が実施され元のデータブロックを示すサンプルポイントの縮小サイズのマトリックスを生成する。補間処理の場合は、零値の追加周波数成分が周波数成分アレーに挿入され、その後の逆変換によりスペクトル帯域幅の増加無しに一連の拡大データサンプルを生成する。周波数領域内

でのデータ及びフィルタカーネルの変換の乗算を伴う畳込み又はフィルタ処理により実現される整形処理の場合は、処理済みデータサンプルの一連のブロックを発生させる逆変換が提供される。カーネルの空間記号は、線形位相フィルタに対して成分数を減少させることにより変更され、零挿入されてデータブロックのサンプル数と同じになる。これに続いて、零挿入されたカーネルマトリックスの離散奇数コサイン変換(D O C T)が形成される。

【0010】米国特許第 5, 175, 617 号明細書には、電話回線帯域制限アナログチャンネルを介してログマップビデオ画像を送信するシステム及び方法が開示されている。ログマップ画像内の画素構成は、中心に画素がより集中する人間の目のセンサの幾何学的配置に一致するようになっている。この送信機は周波数帯をチャンネルに分割し、1 個又は 2 個の画素を各チャンネルに割り当てる。例えば、3 K H z の音声品質電話回線は、それぞれ 3.9 H z の間隔を持つ 768 個のチャンネルに分割される。各チャンネルは直交位相の 2 個の搬送波から成るので、各チャンネルは 2 個の画素を運ぶことが可能である。幾つかのチャンネルは、受信機が受信信号の位相と大きさの両方を検出できるようにする特殊校正信号用に確保されている。センサと画素が発振器部に直結され、受信機が各チャンネルを連続して受信できる場合には、この受信機は送信機と同期をとる必要が無い。F F T アルゴリズムは、受信機が第 1 のフレームと同期をとり、その後フレーム期間毎に後続のフレームを得る連続動作の場合に対する高速離散近似を実行する。フレーム期間はサンプリング期間と比べて比較的低いので、一旦第 1 フレームが検出されると、受信機がフレーム同期を失うことはありそうもない。試験的ビデオ電話では、1 秒に 4 個のフレームを送信し、1440 個の画素ログマップ画像に対し直交符号化を実施し、秒速 40,000 ビット以上の有効データ転送速度が得られた。

【0011】米国特許第 5, 185, 819 号明細書には、奇数及び偶数フィールドのビデオ信号がフレーム内圧縮モードとフレーム間圧縮モードで順番に独立して圧縮された奇数及び偶数フィールドのビデオ信号を有するビデオ圧縮システムが開示されている。独立して圧縮されたデータの奇数及び偶数フィールドは、フレーム内偶数フィールドが圧縮されたデータはフレーム内奇数フィールドが圧縮された連続するフィールド間の途中で発生するように送信のためにインターリーブされる。受信機にとっては、インターリーブされた一連のフィールドにより、送信データ量を増やすことなく符号化のための信号への進入ポイント数が 2 倍になる。

【0012】米国特許第 5, 212, 742 号明細書には、リアルタイムで圧縮／復号のためにビデオデータを処理する装置及び方法が開示されている。この装置は複数の計算モジュールから成り、好ましい実施例では、合

計4個の計算モジュールが並列に接続されている。個々の計算モジュールは、処理装置と、デュアルポートメモリと、スクラッチパッドメモリと、アービトレーション機構で構成されている。第1のバスはこれらの計算モジュールとホストプロセッサを接続する。最後に、この装置はホストプロセッサと計算モジュールに第2のバスを介して接続された共用メモリを有する。この方法は、各処理装置が処理するために画像の部分割当てを行う。

【0013】米国特許第5,231,484号明細書には、提案されているISO/IECMPEG基準の使用に10 適している開示される。これには、協働する3個の構成要素又はサブシステムが含まれる。その画像に割り付けられたビット数を考慮して最適の視覚品質を提供するために、この3個の構成要素又はサブシステムは入力されるデジタル動画列を可変順応的に前処理し、ビットを画像に順番に割り付けし、変換係数を画像の異なる領域にビデオ順に適応的に量子化する。

【0014】米国特許第5,267,334号明細書には、コンピュータシステムでの動画列のフレーム冗長度を20 除去する方法が開示されている。この方法は動画列の第1番目の場面変化を検出する工程と、第1画像にとって完全な場面情報を含む第1キーフレームを発生する工程から成る。好ましい実施例に於いては、この第1キーフレームは「前向き」キーフレーム又はイントラフレームとして知られており、通常CCTT準拠圧縮ビデオデータ内に存在している。この処理は少なくとも1個の中間圧縮フレームを発生する工程を具備し、該少なくとも1個の中間圧縮フレームは、前記動画列内で時間的に第1画像に続く少なくとも1個の画像に対して第1画像との差情報を含む。この少なくとも1個の画像はインタ20 ーフレームとして知られている。最後に、動画列における第2場面変化を検出する工程と、第2場面変化の直前に表示される画像にとって完全な場面情報を含む第2キーフレームを発生する工程がある。この第2キーフレームは「後向き」キーフレームとして知られている。前記第1キーフレームと少なくとも1個の中間圧縮フレームは順方向再生のために連結されており、前記第2キーフレームと前記中間圧縮フレームは逆方向再生のために連結されている。画像が順方向再生される時には、前記イントラフレームは、完全場面情報を発生するのに利用さ40 れることもある。この場面が逆再生される時には、完全場面情報を再生するのに後方向キーフレームが使用される。

【0015】米国特許第5,276,513号明細書には、所定数の公知の画像・ピラミッドステージから成る第1回路装置が、前記所定数と同数の新規な動画・ベクトルステージから成る第2回路装置と共に開示されている。これらの装置は費用効果の高い階層的動画分析(HMA)をリアルタイムで、最少システム処理遅延及び/又は最少システム処理遅延を利用して及び/又は最少ハ50

ードウェア構造で行う。特に、比較的高いフレーム速度(例えば、秒速30フレーム)で発生する連続的所定画素濃度画像データフレームの継続入力列からの比較的高解像度画像データに応答して、前記第1及び第2回路装置は、ある処理システム遅延後に、同じフレーム速度で発生する連続的所定画素濃度ベクトルデータの継続出力列を得る。各ベクトルデータフレームは各対の連続画像フレーム間で起こる画像の動きを示している。

【0016】米国特許第5,283,646号明細書には、リアルタイムビデオ符号化システムに画像を一度だけ符号化しながら、フレーム毎に所望数のビットを精確に送出させ、例えば通信チャンネルを介して送信される画像を表わす係数を量子化するのに使用される量子化ステップサイズを更新する方法と装置が開示されている。このデータは、複数のブロックを有するセクタに分割される。このブロックは、例えばDCT符号化を用いて符号化され、各ブロック毎に係数列を発生する。この係数は量子化され、データを記述するのに要するビット数は量子化ステップに依っては、かなり変動する。各セクタのデータの送信の終了時には、この特定グループのデータと関連する選ばれた数のセクタに対して、使用された実際の累積ビット数は、使用される所望の累積ビット数と比較される。本システムは、例えば画像を表わす複数のセクタに対して最終的所望のデータビット数を得るために、量子化ステップサイズを再調整する。量子化ステップサイズを再新して所望のビット割り付けを決定するため様々な方法が開示されている。

【0017】ウエスコン技術論文(Wescon Technical Papers)No.2、1984年10月/11月号のヨンM. チョン(Chong, Yong M)著の論文、「デジタル画像処理のデータフローアーキテクチャ」(A Data-Flow Architecture for Digital Image Processing)には、画像処理に特に設計されたリアルタイムでの信号処理システムが開示されている。具体的には、固定長のアドレスフィールドを有する固定の1ワード長であるトークンをベースにしたデータフローアーキテクチャが開示されている。トークンはデータフィールド、制御フィールド、およびタグから構成されている。トークンのタグフィールドは、更に、プロセッサアドレスフィールドおよび識別子フィールドとに分けられる。プロセッサアドレスフィールドは、トークンを正しくデータプロセッサに送出するために用いられ、識別子フィールドは、データプロセッサにどのような処理を行うべきかを知らされるためにデータにラベルを付ける目的に用いられる。この様にして、識別子フィールドは、データプロセッサに対しての指示として作用する。システムは各トークンを、モジュール番号(MN)を使って特定のデータフロープロセッサに送る。このMNが特別なステージのMNと一致する場合は、適切な演算をデータに対して行う。もし一致しない場合は、トークンは出力データバスに送られる。

【0018】IEEE J 固体回路(Solid-State Circuits)のVol. 23, No. 1, 1988年2月号の、キモリ(Kimori)他著の「自己調整回路による弾性パイプライン機構」(An Elastic Pipeline Mechanism by Self-Timed Circuits)では、自己調整回路を有する弾性パイプラインが開示されている。非同期のパイプラインは複数のパイプラインステージを具備する。各パイプラインステージは、パイプラインステージ特有の論理演算を行う組み合わせ論理回路に続いて設けられている。入力データラッチグループにより構成される。データラッチには、このパイプラインステージに伴うデータ転送制御回路から生成されるトリガ信号が同時に供給される。データ転送制御回路は相互に連結してチェーンを形成し、これを介して送信信号線並びに肯定応答信号線により、後段のパイプライン間のデータ転送をハンドシェイクモードに制御している。また、現在のパイプラインステージのオペランドに対して行う演算を選択するために、通常デコーダが各ステージに設けられている。更に、複雑な復号処理を予めコード処理するために、また、論理回路にとって重大な1経路の問題を軽減するために、前段にデコーダを配置することもできる。サブモジュール間の相互作用は完全に局所化した決定に基づき決められ、また、各サブモジュールが独自にデータバッファリング及び自己調整データ転送を同時にデータ転送を同時に行えるので、パイプラインの柔軟性により、いかなる集中制御も除去される。最後に、パイプラインの柔軟性増すために、空のパイプラインステージを、占有されているパイプラインステージ間に設け、各ステージ間での信頼性のあるデータ転送を確保している。

【0019】

【発明が解決しようとする課題】本発明は、入力、出力、入力と出力との間の複数の処理ステージとを具備し、複数の処理ステージはトークンをパイプラインに沿って伝送するために2線インターフェースによって相互接続されており、さらに本発明は処理ステージの間でデータ機能、および/または組み合わせられた制御-データ機能を制御するためにパイプライン内の全ての処理ステージと、あるいは選択された処理ステージのみとインターフェースするために万能適応ユニットの形態の制御、および/又はデータトークンを具備し、その結果、パイプライン中の処理ステージは構成及び処理において強化された柔軟性が与えられる。本発明によれば、処理ステージは少なくとも1つのトークンの認識に回答して構成可能である。処理ステージの1つは入力を受信しトークンを発生、および/または変換するスタートコード検出器である。

【0020】さらに、本発明は、空間デコーダシステムを具備し、ビデオデータ用であり、ハフマンデコーダ、インデックス/データユニット、演算論理ユニット、複数の異なるピクチャ圧縮/伸長規格の各々に対する記憶

された個別プログラムを持つマイクロコードROMを具備し、プログラムはトークンによって選択可能であり、複数のピクチャ規格に対する処理が容易になる改良されたパイプラインシステムに関する。

【0021】

【課題を解決するための手段】簡単に且つ一般的に、本発明は、入力と、出力と、この入力と出力間にある複数の処理ステージを有するパイプラインシステムにおいて、処理ステージはパイプラインに沿ってトークンを伝送するために2線インターフェースにより相互接続され、制御トークン、データトークンはパイプラインの中の全ての処理ステージとインターフェースし、パイプラインの中の選択された処理ステージと相互作用するユニバーサル適応ユニットの形態であり、処理ステージは構成と処理において向上した柔軟性を与えられている。

【0022】パイプラインの中の各処理ステージは第1次、第2次記憶手段の両方を有していてもよい。処理ステージは選択されたトークンに認識に応じて再構成される。パイプラインのトークンはダイナミックに適応的であり、機能を実行する処理ステージに応じて位置するか、あるいは機能を実行する処理ステージに無関係に位置する。

【0023】本発明のパイプラインにおいては、トークンはステージとインターフェースすることにより変更されることもある。トークンはパイプラインの全てのステージと相互作用することもあるし、あるいは全てよりは少ない特定のステージのみと相互作用する場合もある。パイプライン中のトークンは隣接するステージと相互作用することもあるし、あるいは隣接しないステージと相互作用する場合もある。トークンは処理ステージを再構成する。そのようなトークンはある機能に従属して位置することもあるし、他の機能とは無関係に位置することもある。

【0024】トークンは、再構成可能な処理ステージとともに、パイプラインシステムにとっての基本的なビルディングブロックを提供する。トークンとパイプラインの処理ステージとの相互作用は当該処理ステージの以前の処理ヒストリーによって条件付けられる。トークンはそのトークンを特徴付けるアドレスフィールドを有しても良い。これにより、処理ステージとの相互作用はこのアドレスフィールドで決定される。

【0025】トークンは更に、このトークンに追加のワードが存在することを示し、該トークン内の最終ワードを識別する拡張ビットを含んでも良い。トークンのアドレスフィールドは可変長でも良いし、典型的にはホフマン符号化されている。

【0026】トークンは処理ステージにより発生される。パイプライントークンは処理ステージに転送するデータを含んでも良いし、トークンはデータが無くても良い。あるトークンはデータトークンとして識別さ

れ、パイプラインの処理ステージにデータを与える。他のトークンは制御トークンとして識別され、処理ステージを調整する。この調整は処理ステージの再構成を含む。さらに、他のトークンはデータの供給と調整の両方を処理ステージに対して実施する。トークンのいずれかはパイプラインの処理ステージに対する符号化標準を識別する。他のトークンは処理ステージ間でどんな符号化標準にも依存せず動作する。トークンは処理ステージによる連続する変更も可能である。

【0027】処理ステージと協働してのトークンの相互作用の柔軟性は内部構造に対する処理ステージの大きな機能分散を容易にし、トークンの柔軟性はシステム拡張及び/又は変更を容易にする。この点について、トークンは処理ステージ内の複数の機能を容易にすることもできる。トークンはハードウェアに基づいてもソフトウェアに基づいても良い。トークンは、データと制御を同時に処理ステージに提供にする。

【0028】本発明によれば、単一のビットストリーム中に伝達され、制御コードと対応するデータとがそれぞれ符号化された対を有し、デジタルビットのシリアルビットストリームとしての複数の符号化ビットストリームを扱うための、2線インターフェースにより複数の処理ステージが相互接続されたパイプラインシステムにおいて、単一のシリアルビットストリームに応答し制御トークンとデータトークンを発生し、2線インターフェースに与えるスタートコード検出手段と、所定の処理ステージに設けられ、トークンを該ステージに特有な制御トークンとして認識し、パイプラインに沿って不認識の制御トークンを通過させるトークンデコード手段と、識別されたデータトークンを扱うために、認識された制御トークンに応答し特定のステージを再構成する再構成デコード・パーザ処理手段とを具備することを特徴とする。

【0029】また、本発明によれば、第1、第2のレジスタを具備し、第1のレジスタはデコード・パーザ手段の入力に接続され、第2のレジスタはデコード・パーザ手段の出力に接続され、処理ステージのいずれかは空間デコード手段であり、処理ステージの他は制御トークンとデータトークンを発生し2線インターフェースを通過させるトークン発生手段であり、トークンのいずれかを空間デコードに特有な制御トークンとして認識し、制御トークンに続くデータトークンを空間的にデコードするために空間デコードを第1の復号化フォーマットに構成するトークンデコード手段が空間デコード手段に設けられている。

【0030】さらなる処理ステージはパイプラインの空間デコードの下流に配置される時間デコード手段であり、時間デコード内に位置する第2のトークンデコード手段は所定のトークンを時間デコードに特有の制御トークンとして認識し、制御トークンに続くデータトークン

を時間的に復号化し時間デコードを第1の復号フォーマットに構成することを特徴とする。時間デコードは予測トークンによって再構成可能な再構成予測フィルタを利用することができる。

【0031】8×8画素ブロックのデータが時間デコードの中を2線インターフェースに沿って移動し、ブロックの境界に沿ったブロックを格納し検索するアドレス手段が設けられている。アドレス手段はブロック境界を横切るデータブロックを格納、検索することも可能である。アドレス手段はデータブロックを表示のための画素データとして再配列してもよい。格納、検索されるデータブロックは8×8画素ブロック以外のサイズでもよい。時間デコードの出力を表示する手段、あるいは出力を画素メモリに再書き込みする手段を設けてもよい。復号化フォーマットは静止画フォーマット、動画フォーマットのいずれでもよい。

【0032】処理ステージは、トークンのアドレスをデコードするトークンデコードと、処理ステージのコンフィグレーションを実行するためにトークンデコードに回答するアクション識別部とを具備する。処理ステージは、2線インターフェースを介して相互接続された複数の処理ステージを有するパイプライン処理マシンに存在し、制御トークンとデータトークンが2線インターフェースを通過する。トークンデコード回路は処理ステージのある位置に配置され、あるトークンを該ステージに特有の制御トークンとして認識し、認識されないトークンをパイプラインに沿って伝達させる。2線インターフェース上の処理ステージに先行する位置に第1の入力ラッチ回路が設けられ、処理ステージに後続する位置に第2の入力ラッチ回路が設けられる。トークンデコード回路は第1の入力ラッチ回路を介して2線インターフェースに接続される。所定の処理ステージは所定のデータ格装置の出力に接続されるデコード回路を具備し、各処理ステージは、ステージが所定のステージ動作化信号を含む時、所定のステージ非動作化パターンを含むまで動作化モードである。

【0033】本発明は、デジタルピクチャ情報処理システムにおいて、複数の異なるピクチャ圧縮/伸長規格に応じてデータを処理できるようにシステムを選択的に構成する手段を提供する。ピクチャ規格は、JPEG、MPEG、H. 261、他の規格、あるいは本発明の要旨を逸脱しない範囲でこれらの規格を適宜組み合わせた規格である。さらに、本発明は、システムは、空間デコードシステムを具備し、ビデオデータ用であり、ハフマンデコード、インデックス/データユニット、演算論理ユニット、複数の異なるピクチャ圧縮/伸長規格の各々に対する記憶された個別プログラムを持つマイクロコードROMを具備し、プログラムはトークンによって選択可能であり、複数のピクチャ規格に対する処理が容易になる。多重規格システムは、選択したピクチャ規格に無

関係にトークンの動作を利用することができる。トークンは種々のピクチャ規格の全てについてシステムの一般的な通信プロトコルとして利用することができる。本システムは、さらに、単一のシリアルデータストリーム上に配列された異なる符号化データストリームを規格に従属する、あるいは規格と独立するハードウェアと制御トークンとを用いて単一のデコーダにマッピングする多重規格トークンを特徴とする。システムは異なるピクチャ規格に関連するマクロブロックを共通のアドレッシング法に配置するアドレス発生手段を具備する。

【0034】以下の本発明の実施例の記載中、下記の用語が頻繁に使用されるので、以下にその用語の一般的説明をする。

【0035】用語説明

ブロック：8行8列の画素のマトリックス、又は64DCT係数（ソース、量子化又は逆量子化）。

【0036】クロミナンス（成分）：ビットストリームで規定されるような三原色に係わる2個の色差信号の1個を表わす1つのマトリックス、ブロック、又は1個の画素。色差信号に使用される符号はCrとCbである。

【0037】符号化表記：符号化形式で表わされたデータエレメント。

【0038】符号化ビデオビットストリーム：本明細書で規定しているところの、一連の1個以上の画像の符号化表記。

【0039】符号化順番：画像が送信され符号化される順番。この順番は必ずしも表示順番と同一である必要はない。

【0040】成分：1つのマトリックス、ブロック、又は画像を構成する3つのマトリックス（輝度と2個のクロミナンス）の内の1つからの1個の画素。

【0041】圧縮：1項目のデータを表わすのに使用されるビット数を減少すること。

【0042】デコーダ：復号化処理の実施例。

【0043】復号化（処理）：本明細書での定義では、入力された符号化ビットストリームを読みだし復号化された画像又は音声サンプルを発生する処理。

【0044】表示順番：復号化画像が表示される順番。一般的には、これは画像がエンコーダに入力された順番と同じである。

【0045】符号化（処理）：本明細書では特定されていないが、入力画像列又は音声サンプルを読みだし、本明細書で定義されている有効な符号化ビットストリームを発生する処理。

【0046】イントラ(intra)符号化：マクロブロック又は画像からの情報だけを使用するマクロブロック又は画像の符号化。

【0047】輝度（成分）：信号の白黒を表わす1つのマトリックス、ブロック、又は1個の画素で、ビットストリームで規定されるような三原色に係わるもの。輝度

信号に使用される符号はYである。

【0048】マクロブロック：4つの、8×8ブロックの輝度データと、画像の輝度成分の16×16部分からの2つ（4：2：0クロマフォーマット用）、4つ

（4：2：2クロマフォーマット用）又は8つ（4：4：4クロマフォーマット用）の対応する8×8ブロックのクロミナンスデータ。マクロブロックは画素データを言う場合もあり、画素値と、本明細書のこの部分に定義されているシンタックスのマクロブロックヘッダに規定されているその他のデータエレメントの符号化されたものを言う場合もある。通常の知識を有する当業者にとって、その使用は文脈からも明らかである。

【0049】動画圧縮：動画ベクトルを使って画素値の予測効率を向上させること。この予測は動画ベクトルを使用し、予測エラー信号の作成に使用される、以前に復号化された画素値を含む過去の及び／又は未来の基準画像にオフセットを提供する。

動画ベクトル：現在の画像の座標位置から基準画像の座標へのオフセットを提供する動画補償に使用される2次元ベクトル。

【0050】非イントラ(non-intra)符号化：マクロブロック又は画像それ自体からの情報と別の時刻に発生したマクロブロックと画像からの情報も使用するマクロブロック又は画像の符号化。

【0051】画素：画像単位

ピクチャ：ソース、符号化又は再構成された画像データ。ソース又は再構成された画像はそれぞれ輝度と2個のクロミナンス信号を表わす8ビットの数値を有する3個の四角マトリックスから成る。先進のビデオでは、画像はフレームと同じであり、インターレースされたビデオでは、文脈により画像はフレーム、又はそのフレームの先頭のフィールド又は最後のフィールドを指す。

【0052】予測：予測器を使用して現在復号化されている画素値又はデータエレメントを予測すること。

【0053】再構成可能処理ステージ(RPS)：認識されたトークンに応答して、自己を再構成し各種作業をするステージ。

【0054】スライス：一連のマクロブロック。

【0055】トークン：制御及び／又はデータ機能のための相互作用インターフェースメッセージのパッケージの形式のユニバーサル適応ユニット。

【0056】開始コード（システム及びビデオ）：ユニークな符号化ビットストリームに挿入された32ビットの符号。これらの符号は、符号化シンタックス内の構造の幾つかを識別することを含めた幾つかの目的に使用される。

【0057】可変長符号化(VLC)：より短いコード語を頻繁なイベントに割り当てる、あるいはより長いコード語を頻繁ではないイベントに割り当てる符号化のための逆処理。

【0058】ビデオシーケンス：一連の1つ以上の画像。

【0059】

【作用】本発明によれば、入力、出力、前記入力と前記出力との間の少なくとも1つの処理ステージとを具備するシステムにおいて、処理ステージが制御機能及び／又はデータ機能を確立するための万能適応ユニットの形態の少なくとも1つのトークンの認識にตอบสนองして構成可能であり、処理ステージは構成及び処理において強化された柔軟性が与えられることを特徴とするシステムが提供される。

【0060】

【実施例】以下、図面を参照して本発明によるパイプラインシステムの実施例を説明する。

【0061】本発明の好ましい実施例に用いられるパイプラインシステムに使われる最も一般的な機能に関する概説として、図1は、6ステージパイプラインの6サイクルの大幅に簡素化された説明図である。（以下に更に詳細に説明するように、パイプラインの好ましい実施例には、図1には示されない幾つかの有利な機能が含まれる。）図面を参照することとし、図面に含まれる種々の図を通じて、同じ参照番号は同じか又は対応するエレメントを示し、そして、更に詳細には、図1は、本発明を実施した場合の6サイクルのブロックダイアグラムである。ボックスの各列は1つのサイクルを示し、そして、異なるステージの各々はそれぞれAからFまでに分類される。斜線を引いた各ボックスは、対応するステージが有効なデータ、即ち、パイプラインステージの1つにおいて処理されるべきデータを保持することを示す。処理（データの操作なしの単純な転送以外の何も含まれないこともあり得る）の後で、有効なデータは、有効出力データとしてパイプラインから転送される。

【0062】実際のパイプラインアプリケーションには、6つよりも多いか又は少ないパイプラインステージが含まれることがあることに注意されたい。当然気が付くように、本発明は、任意の個数のパイプラインステージの場合に使用可能である。更に、データは、1つよりも多いステージにおいて処理しても差し支えなく、そして、ステージが異なれば処理時間も異なることがあり得る。

【0063】クロック及びデータ信号（以下に説明する）に加えて、パイプラインのは、2つの転送制御信号、即ち、「VALID」（有効）信号及び「ACCEPT」（許容）信号が含まれる。これらの信号は、パイプライン内のデータ転送を制御するために使われる。隣接するステージを接続する2本のラインのうちの上側のラインとして図示されるVALID信号は、各パイプラインステージから最寄りの隣接デバイスまで順方向すなわち下流方向に供給される。このデバイスは、他のパイプラインステージ、または或る別のシステムであっても

差し支えない。例えば、最後のパイプラインステージは、そのデータを次の処理回路構成に供給しても差し支えない。隣接するステージを接続する2本のラインのうちの下側のラインとして図示されるACCEPT信号は、もう一方の上流方向に向かって前のデバイスまで供給する。

【0064】本発明の実現に使用されるタイプ of データパイプラインシステムは、好ましい実施例に示すように、次に示す特性の1つかそれ以上の特性を持つ。

【0065】1. パイプラインは「弾力性がある」、即ち、特定のパイプラインステージに起きた遅延に起因して他のパイプラインステージに引き起こされる外乱は可能な限り小さな外乱である。連続したパイプラインステージは処理を継続することが可能であり、従って、遅延したステージに後続するデータの流れには間隙を生じることを意味する。同様に、先行するパイプラインステージも、出来る限り継続可能である。この場合、データストリーム中の間隙は、出来る限り、データの流れから除去される。

【0066】2. パイプラインを調停する制御信号は、これらの信号は最寄りの隣接パイプラインステージに限り伝播するように組織される。データの流れと同じ方向に流れる信号の場合には、前記のステージは直ぐ後に続くステージである。データの流れと逆方向へ流れる信号の場合には、前記のステージは直ぐ前のステージである。

【0067】3. パイプライン内のデータは、多数の異なるタイプのデータが、当該パイプライン内で処理されるようにコード化される。この符号化は、可変サイズのデータパケットを収容し、そして、パケットのサイズは、前もって既知である必要はない。

【0068】4. データのタイプ記述と関連するオーバーヘッドはできる限り小さい。

【0069】5. パイプラインステージに要求される機能にとって必要な最小個数のデータタイプのみを認識することが各パイプラインステージにとって可能である。ただし、パイプラインステージは、前記のデータタイプを認識しない場合であっても、これら全てのデータタイプを連続したステージに供給することが可能でなくてはならない。これが可能であれば、隣接しないパイプラインステージ間の通信を可能にする。

【0070】図1には図示されていないが、データラインが有り、複数本の単線または数本の平行線のいずれであっても、各パイプラインステージに対して導入または導出するデータバスを形成する。以下に極めて詳細に説明および図解するように、データは、パイプラインのステージに対して、データラインを介して、転入、転出、及びステージ間で転送される。

【0071】第1のパイプラインステージは、任意の形の先行デバイスからのデータ及び制御信号を受信可能で

あることに注意されたい。この場合の先行デバイスとは、例えば、デジタルイメージ送信システムの受信回路、他のパイプライン等を意味する。他方において、第1のパイプラインステージは、当該パイプラインにおいて処理されるデータの全部または一部を生成可能である。実際問題として、以下に詳しく説明するように、

「ステージ」は、何もしないシステム（単にデータを提供するだけ）又は全システム（例えば、他のパイプライン、または多重システム或は多重パイプライン）を含む任意の処理回路を含んでも差し支えなく、そして、「ステージ」は、必要に応じて、データを生成、変更、及び削除可能である。パイプラインステージが当該パイプラインを通過して下方に転送される有効なデータを含む場合、データ妥当性を示すVALID信号は、直ぐ後続する次のパイプラインステージより更に速くに転送される要がない。従って、2線インターフェースは、当該システムの対を構成する全てパイプラインステージの間に含まれる。即ち、いわゆる他のデバイスが含まれ、そして、データが、この種デバイスとパイプラインとの間で転送される場合には、先行デバイスと第1ステージとの間、及び後続デバイスと最後のステージとの間の2線インターフェースが含まれる。信号の各々、即ち、ACCEPT、及びVALIDは、HIGH、及びLOWの値を持つ。これらの値は、各々、「H」、及び「L」と略記される。本発明を実現する場合、パイプラインの最も一般的なアプリケーションは、通常、デジタルである。この種のデジタルとして実現した場合、例えば、HIGH値は論理的「1」であり、そして、LOW値は論理的「0」であっても差し支えない。ただし、システムは、デジタルとして実現するよう制限される訳ではなく、アナログとして実現した場合、HIGH値は、或る電圧、または他の類似の設定スレシールド以上の量であっても差し支えなく、LOW値は、同一又は他のスレシールド以下（或いは以上）の対応する信号によって示されても差し支えない。デジタルアプリケーションの場合には、本発明は、例えば、CMOS、バイポーラ等のようなあらゆる既知の技術を用いて実現することが可能である。

【0072】VALID信号の記憶装置を提供するために個別の記憶そうち及びワイヤを使用する必要はない。デジタル実施例においてこれは真である。データの「妥当性」の指示がデータと共に記憶されることだけが必要とされる全てである。単に一例として挙げると、デジタル値によって表されるデジタルテレビ画像においては、国際規格CCIR 601に規定されているように、ある特定の値は許容されていない。このシステムにおいては、画像のサンプルを表すために8ビット2進数が使用され、そして、ゼロ及び255の値は使用してはならない。

【0073】この種の画像が、本発明の具体化に組み込

まれたパイプラインにおいて処理される必要がある場合には、これらの値の1つ（例えば、ゼロ）は、パイプライン内の特定のステージにおけるデータが有効でないことを示すために使われることになる。従って、あらゆるゼロでないデータは有効であるとみなされるべきである。この例においては、識別可能であって、しかも、関連しているデータの「有効性」を記憶しつつあると言うことのできる特定のラッチは無い。それにも拘わらず、データの妥当性は、データと共に記憶される。

10 【0074】図1に示すように、各ステージへのVALID信号の状態は、上側の右向き矢印上において「H」、または「L」として示される。従って、ステージAからステージBへのVALID信号はLOWであり、そして、ステージDからステージEへのVALID信号はHIGHである。各ステージへのACCEPT信号の状態は、下側の左向き矢印上に「H」、または「L」として示される。従って、ステージEからステージDへのACCEPT信号はHIGHであり、一方、ステージFへのパイプラインの下流に接続されたデバイスからのACCEPT信号はLOWである。

20 【0075】上流の隣接ステージから下流のステージへ向かうACCEPT信号がHIGHである場合にはいつでも、データは、1サイクル期間中に1つのステージからもう一方のステージまで転送される（以下に説明する）。2つのステージ間におけるACCEPT信号がLOWである場合には、データは、これらのステージの間で転送されない。

30 【0076】再び図1を参照して、1つのボックスに陰がつけられている場合には、対応するパイプラインステージは、例えば、有効な出力データを含むものと仮定される。同様に、当該ステージから次のステージまで供給されるVALID信号はHIGHである。ステージB、D、及びEが有効なデータを含む場合におけるパイプラインを図1に示す。ステージA、C、及びFは、有効なデータを含まない。開始点においては、パイプラインステージAへのVALID信号はHIGHであり、パイプラインへの伝送回線上のデータが有効であることを意味する。

40 【0077】同様に、この時点において、パイプラインステージFへのACCEPT信号はLOWであり、従って、有効であると有効でないに拘わらず、ステージFからデータは転送されない。有効および無効データがパイプラインステージ間において転送されることに注意されたい。価値のないデータを保管する無効データは重ね書きされ、それにより、無効データをパイプラインから除去する。ただし、有効データは処理用または例えば、パイプラインステージ、当該パイプラインからのデータを受け取るパイプラインに接続されたデバイスまたはシステムのような下流のデバイスにおいて使用するために保管されなければならないので、有効データは重ね書き

されてはならない。

【0078】図1に示すパイプラインにおいて、ステージEは有効なデータD1を含み、ステージDは有効なデータD2を含み、ステージBは有効なデータD3を含み、そして、上流パイプラインと接続されたデバイス（図示せず）は、当該パイプラインに転送され、そして、処理されるべきデータD4を含む。上流デバイスに加えて、ステージB、D、及びEは有効なデータを含み、従って、これらのステージまたはデバイスからそれぞれ後続するデバイスへのVALID信号はHIGHである。ただし、ステージA、C、及びFからのVALID信号は、これらのステージが有効なデータを含まない

ので、LOWである。
【0079】パイプラインから下流に接続されたデバイスが、パイプラインからデータを受取入れる準備が整っていないものと仮定する。デバイスは、対応するACCEPT信号をLOWにセットすることにより、この状態をステージFに送信する。ただし、ステージF自体は有効なデータを含まず、従って、先行ステージEからデータを受け入れることができる。従って、ステージFから

ステージEへのACCEPT信号はHIGHにセットされる。
【0080】同様に、ステージEは有効なデータを含み、そして、ステージFは、このデータを受け入れる準備が整っている。従って、有効なデータD1が最初にステージFへ転送される限りステージEは新しいデータを受け入れることができる。換言すれば、ステージFはデータを下流に転送することができないが、他の全てのステージは、あらゆる有効なデータを重ね書き又は失うことなしに下流に転送できる。従って、サイクル1の終点において、データは右へ1のステップだけ「シフトされる」ことが可能である。この状態をサイクル2に示す。

【0081】図に示す例において、下流のデバイスは、サイクル2に新しいデータを受け入れる準備がまだ整っていないので、ステージFへのACCEPT信号はまだLOWである。従って、ステージFは新しいデータを受け入れることが出来ない。理由は、そうすると、有効なデータD1が重ね書きされ、そして、失われるからである。従って、ステージEは同様に有効なデータD2を含むので、ステージEからステージDへのACCEPT信号の場合と同様に、ステージFからステージEへのACCEPT信号はLOWになる。ただし、AからDまでの全てのステージは新しいデータを受け入れることが可能であり（これらのステージが有効なデータを含まないか、或いは、これらのデータは、それらの有効なデータを下流にシフトし、そして、新しいデータを受け入れることができるので）、そして、これらのステージは、それぞれ対応するACCEPT信号をHIGHにセットすることによってこの条件を直前を先行する隣接ステージに送信する。

【0082】サイクル3と行分類された期間中におけるサイクル2の後のパイプラインの状態を図1に示す。一例として、下流のデバイスがステージFから新しいデータを受け入れる準備がまだ整っていないものと仮定する（ステージFへのACCEPT信号はLOWである）。従って、ステージE及びF、まだ「ブロック」されているが、サイクル3においては、ステージDは有効なデータD3受領済みであり、このステージに以前から在った無効のデータの重ね書きが完了している。サイクル3においてステージDはデータD3を供給することができないので、このステージは新しいデータを受け入れることができず、従って、ステージCへのACCEPT信号をLOWにセットする。ただし、ステージA-Cは新しいデータを受け入れる準備が整っており、そして、対応するACCEPT信号をHIGHにセットすることによってこれを送信する。データD4はステージAからステージBにシフトされてしまっていることに注意されたい。

【0083】ここで、下流のデバイスがサイクル4において新しいデータを受け入れる準備が整った状態になると仮定することとする。この下流デバイスは、ステージへのFACCEPT信号をHIGHにセットすることによって、このことを、パイプラインに送信する。ステージC-Fは有効なデータを含むが、これらのステージは、この状態で、データを下流にシフトすることが可能であり、従って、新しいデータを受け入れることが可能である。従って、各ステージはデータを1ステップだけ下流にステップすることができるので、これらのステージは、それぞれ対応するACCEPT信号をHIGHの外にセットする。

【0084】最終のパイプラインステージ（この例ではステージF）へのACCEPT信号がHIGHである限り、図1に示すパイプラインは、硬直パイプラインとして作動し、そして、各サイクルごとにデータを1ステップだけシフトするだけである。従って、サイクル4においてステージFに含まれるデータD1は、サイクル5において、パイプラインから次のデバイスにシフトされ、そして、他の全てのデータは1ステップだけ下流にシフトされる。

【0085】ここで、ステージFへのACCEPT信号がサイクル5においてLOWになると仮定する。これは、再度説明すれば、ステージD-Fが新しいデータを受け入れることができず、そして、これらのステージから直前の先行隣接ステージへのACCEPT信号はLOWになる。従って、データD2、D3、及びD4は、下流にシフト出来ないが、データD5は可能である。従って、サイクル5の後のパイプラインの対応する状態はサイクル6として図1に示される。

【0086】本発明の好ましい実施例に従えば、空の処理ステージを「フィルアップ（充填）」するパイプラインの能力は高度に有利である。理由は、この能力によ

て、パイプラインの処理ステージが相互に結合から解放されるからである。換言すれば、パイプラインステージのデータを受け入れる準備が整っていない場合であっても、パイプライン全体は、停止するか、または遅延ステージを待つ必要がない。そればかりでなく、1つのステージが有効なデータを受け入れることができない場合に、当該ステージは、単に、パイプライン内の時間的な「壁」を形成するに過ぎない。この状態になっても、「壁」の下流のステージは、パイプラインに接続された回路に有効なデータを前進させ続けルコトサエ可能であり、そして、「壁」の左側のステージは、依然として、有効なデータを受け入れ、そして、下流に向かって転送することができる。数個のパイプラインステージが一時的に新しいデータを受け入れることが出来ない場合、他のステージは、正常作動を継続できる。特に、パイプラインは、その次のステージが新しいデータを受け入れる準備が整っていないので前進させることの出来ない有効なデータがステージAにまだ含まれていない限りデータをその最初のステージAに受け入れ続けることが可能である。この例に示すように、1つ又は複数の処理ステージがブロックされた状態においても、データは、パイプライン、及びステージの間に転送可能である。

【0087】図1に示す実施例において、種々のパイプラインステージが、それらの直後に隣接するステージから受け取るACCEPT信号を記憶しないものと仮定する。その代りに、下流ステージへのACCEPT信号がLOWになると、このLOW信号は、有効なデータを含まない最も近いパイプラインステージまで流れをさかのぼって伝播される。例えば、図1を参照して、ステージFへのACCEPT信号がサイクル1においてLOWになるものと仮定する。サイクル2において、LOW信号は、ステージFからステージDに戻って伝播する。

【0088】サイクル3において、データD3がステージDにラッチされている場合、ACCEPT信号は、流れをさかのぼる方向に4つのステージをステージCまで伝播する。サイクル4においてステージFへのACCEPT信号がHIGHになる場合には、この信号は、流れに逆らってステージCまで伝播する。換言すれば、ACCEPT信号の変化は、4つのステージを通して後方に伝播しなければならない。ただし、図1に示す実施例において、新しいデータを受け入れることができる中間ステージがある場合には、ACCEPT信号は、パイプラインの始点まで全経路を後方に伝播する必要はない。

【0089】図1に示す実施例において、各パイプラインステージは、意図的でない重ね書きを実施することなしに、ステージ間でデータを転送可能にするためには、依然として、個別の入力、及び出力データラッチを必要とする。更に、図1に示すパイプラインは「圧縮」可能であるが、下流パイプラインステージがブロックされる場合、即ち、これらのステージが含むデータをこれらの

ステージが供給できない場合、パイプラインは、有効なデータを含むステージの間に有効なデータを含まないステージを提供するために、「膨張」しない。そうではなくて、圧縮する能力は、第1のパイプラインステージに有効なデータが所在しない期間に相当するサイクルに依存する。

【0090】例えば、サイクル4において、ステージFへのACCEPT信号がLOWのままであり、そして、有効なデータによってパイプラインステージA及び3が満たされた場合、有効なデータがステージAに所在し続ける限り、パイプラインは、それ以上圧縮不可能であり、そして、有効な入力データは失われることがある。それにも拘わらず、図1に示すパイプラインは、データが失われる危険を軽減する。理由は、有効なデータを含まないパイプラインステージがある限り当該パイプラインは圧縮可能であることに因る。

【0091】論理的方法において圧縮および膨張の両者共に可能であり、そして、ACCEPT信号の伝播を最も近い先行ステージまでに制限する回路を有するパイプラインの他の実施例を図2、図3に示す。この実施例を実現するための回路について以下に詳しく説明及び図説するが、図2、図3は、この実施例の作動原理を図説するために役立つ。

【0092】比較を容易にするために、図2、図3に示すパイプライン実施例への入力データ及びACCEPT信号は、図1に示すパイプライン実施例の場合と同じである。従って、ステージE、D、及びBは、それぞれ、有効なデータD1、D2、及びD3を有する。ステージFへのACCEPT信号はLOWであり、データD4は、開始パイプラインステージAに供給される。図2、図3には、各近隣の一对のパイプラインステージの各隣接する対になったステージを接続する3本の線が示される。バスであっても差し支えない最上位置の線はデータラインである。中央の線はVALID信号がこれを通して転送されるラインであり、最下位の線は、これを通してACCEPT信号が転送されるラインである。更に、以前の場合と同様に、ステージFへのACCEPT信号は、サイクル4中を除き、LOWのままである。更に、サイクル4において、付加的データD5がパイプラインに供給される。

【0093】図2、図3において、各パイプラインステージは、このパイプラインの実施例における各ステージが一次および第2データ記憶エレメントを含むことを図説するために2つの半分部分に分割された1つのブロックとして表される。図2、図3において、一次データ記憶装置は、各ステージの右半分として示される。ただし、この記述は説明のみ目的とするものであり、このように制限されることを意図したものではないことを理解されたい。

【0094】図2、図3に示すように、ステージへのA

ACCEPT信号がHIGHである限り、データは、任意の所定サイクル期間中に、当該ステージの一次記憶エレメントから、その次のステージの二次記憶装置エレメントに転送される。従って、ステージFへのACCEPT信号はLOWであるが、全ての他のステージへのACCEPT信号はHIGHであり、その結果、データD1、D2、及びD3はサイクル2において1つのステージだけ前方にシフトされ、そして、データD4は、第1のステージAにシフトされる。

【0095】この点までは、図2、図3に示すパイプライン実施例は、図1に示すパイプライン実施例と同様の方法において作動する。ただし、ステージFへのACCEPT信号がLOWであってもステージFからステージEへのACCEPT信号はHIGHである。以下に説明するように、二次記憶装置エレメントがあるために、LOW ACCEPT信号にとって、ステージFを越えて上流まで伝播することは必要ではない。更に、ステージEへのACCEPT信号をHIGHのままに残すことによって、ステージF信号は、新しいデータを受け入れる準備が整っていることを送信する。ステージFは、サイクル3において、その一次記憶エレメント内のデータD1を下流に転送することは出来ない（ステージFへのACCEPT信号はLOWである）ので、従って、ステージEは、データD2を、ステージFの二次記憶装置エレメントに転送する。ステージFの一次および二次記憶装置エレメントは、この状態において、供給できない有効なデータを含むので、ステージFからステージEへのACCEPT信号はLOWにセットされる。従って、これは、LOW ACCEPT信号が、サイクル2に対してただ1ステージだけ後方に伝播することを表し、この場合、このACCEPT信号は、図1に示す実施例におけるステージCまでの全行程を後方に伝播されていなければならない。ステージA-Eは、それらのデータを供給できるので、当該ステージからそれらの直前隣接ステージへのACCEPT信号がHIGHにセットされる。従って、データD3及びD4は右へステージ1つだけシフトされる。その結果、サイクル4において、データは、それぞれステージE及びステージCの一次データ記憶エレメントにロードされる。ステージEは、この段階において、その一次記憶エレメント内に有効なデータD3を含んでいるが、その二次記憶装置エレメントは、一切の有効なデータを重ね書きする危険なしに、他のデータを記憶するために、依然として、使用できる。

【0096】ここにおいて、以前の場合と同様に、ステージFへのACCEPT信号が、サイクル4においてHIGHになると仮定する。このことは、パイプラインがデータを供給する下流のデバイスがパイプラインからデータを受け入れる準備が整っていることを示す。ただし、ステージFのACCEPT信号はLOWにセットされているので、ステージFが新しいデータを受け入れる

準備ができていないことをEステージに示す。各サイクル期間におけるACCEPT信号は、次のサイクルにおいて何が起きるか、即ち、データが供給されるか（ACCEPT HIGH）どうか、或いは、データが所定の場所に残らなければならないか（ACCEPT LOW）どうか、を示すことを考察されたい。従って、サイクル4からサイクル5までの間に、データD1がステージFからその次のデバイスまで供給され、データD2がステージFの二次記憶装置から一次記憶装置までシフトされるが、しかし、ステージEにおけるデータD3はステージFへ転送されない。その次のステージのACCEPT信号はHIGHであるので、データD4及びD5は通常通り、次のパイプラインステージに転送されることが可能である。

【0097】サイクル4及びサイクル5におけるパイプラインの状態を比較することにより、二次記憶装置エレメントを装備することによって、図2、図3に示されたパイプライン実施例が膨張可能となること、即ち、データ記憶エレメントを、その中に有効なデータが前進できるように解放することが分かる。例えば、これらのデータブロックのデータは、ステージFへのACCEPT信号がHIGHになるまで転送出来ないで、サイクル4において、データブロックD1、D2、及びD3は「固体の壁」を形成する。ただし、この信号が一旦HIGHになれば、データD1はパイプラインからシフトされ、データD2はステージFの一次記憶エレメント内にシフトされ、そして、その次のデバイスがデータD2を受け取ることが出来ず、そして、パイプラインがもう一度「圧縮」しなければならない場合、ステージFの二次記憶装置エレメントは自由になって新しいデータを受け入れることができる状態になる。これをサイクル6に示す。このサイクルの期間中に、データD3は、ステージFの二次記憶装置エレメント内にシフトされ、そして、データD4は、通常通り、ステージDからステージEに供給される。

【0098】図4、図5、図6、図7は、パイプラインの好ましい実施例を全体的に示す。この好ましい実施例は、位相が $\phi 0$ および $\phi 1$ の2相非重複クロックを用いて、図2、図3に示す構造を実現する。2相クロックが好ましいが、本発明の様々な実施例を、2相以上のクロックを用いてドライブすることも可能であることを理解されたい。

【0099】図4、図5、図6、図7に示すように、各パイプラインステージは、一次および二次記憶装置エレメントを示す2つの個別ボックスを持つものとして図示される。更に、VALID信号及びデータラインは、以前の場合と同様に、様々なパイプラインステージを接続するが、説明を容易にするために、図4、図5、図6、図7にはACCEPT信号のみを示す。或るACCEPT信号の1つのクロック位相中における状態の変化は、

LOWからHIGHへの変化は上向き矢印を用いて図4、図5、図6、図7に示される。同様に、HIGHからLOWへの変化は、下向き矢印で示される。1つの記憶エレメントからもう1つの記憶エレメント一方へのデータの転送は、大きい開いた矢印によって示される。任意の所定ステージの一次或いは二次記憶エレメントからのVALID信号は、記憶エレメントが有効なデータを含む場合には必ずHIGHであるものと仮定される。

【0100】図4、図5、図6、図7において、各サイクルは、非重複クロック位相 $\phi 0$ 及び $\phi 1$ の1つの全周期で構成されるものとして示される。以下に更に詳細に説明されるように、データは、クロックサイクル $\phi 1$ の期間中に、二次記憶エレメント（各ステージにおける左のボックスとして示す）から一次記憶エレメント（各ステージにおける右のボックスとして示す）に転送され、他方において、データは、クロックサイクル $\phi 0$ の期間中に、1つのステージの二次記憶エレメントからその次のステージの一次記憶エレメントに転送される。同様に、図4、図5、図6、図7は、各ステージにおける一次および二次記憶エレメントは、ACCEPT信号がステージからステージへ供給されると同じ方法においてACCEPT信号を供給するように内部受入れラインを経てさらに接続されることを示す。このようにして、二次記憶エレメントは、そのデータを一次記憶エレメントにいつ供給出来るかを知るはずである。

【0101】図4、図5、図6、図7は、サイクル1の $\phi 1$ 位相を示す。このサイクル中に、それぞれステージE、D、及びBの二次記憶エレメントに既にシフトされているデータD1、D2、及びD3はそれぞれのステージの一次記憶エレメントにシフトされる。従って、サイクル1の $\phi 0$ 位相中における、パイプラインのコンフィギュレーションは、図2のサイクル1の場合と同じである。以前と同様に、ステージFへのACCEPT信号はLOWであるものと仮定される。ただし、図4、図5、図6、図7に示すように、ステージFの一次記憶エレメントへのACCEPT信号はLOWであるが、この記憶エレメントは有効なデータを含まないで、この記憶エレメントはACCEPT信号をその二次記憶エレメント内にHIGHにセットする。

【0102】ステージFの二次記憶エレメントは有効なデータを含まないで、ACCEPT信号は、ステージFの二次記憶エレメントからステージEの一次記憶エレメントへの中に同じくHIGHにセットされる。以前と同様に、ステージFの一次記憶エレメントはデータを受け入れることができるので、全てのの上流一次および二次記憶エレメントにおけるデータは、一切のデータが重ね書きされることなしに、下流にシフト可能である。1つのステージからその次のステージまでのデータのシフトは、サイクル2における次の $\phi 0$ 位相期間中に行われる。例えば、ステージEの一次記憶エレメントに含まれ

る有効なデータD1は、ステージFの二次記憶エレメントにシフトされ、データD4はパイプライン内、即ち、ステージAの二次記憶エレメント内にシフトされる、等々。

【0103】ステージFの一次記憶エレメントは、サイクル2における $\phi 0$ 位相期間中には有効なデータをまだ含まず、従って、一次記憶エレメントからステージFの二次記憶エレメントへのACCEPT信号はHIGHのままである。従って、サイクル2における $\phi 0$ 位相期間中、データは、更に1ステップだけ右へ、即ち、各ステージ内の二次記憶エレメントから一次記憶エレメントまでシフト可能である。ただし、有効なデータが、一度、ステージFの一次記憶エレメントにロードされ、下流デバイスからのステージFへのACCEPTがまだLOWである場合には、有効なデータD1を重ね書きおよび破壊することなしに、ステージFの二次記憶エレメントからデータをシフトすることは出来ない。従って、一次記憶エレメントからステージFの二次記憶エレメントへのACCEPT信号はLOWになる。ただし、二次記憶装置は有効なデータを含まず、そして、そのACCEPT信号出力はHIGHであったので、データD2は、ステージFの二次記憶装置にまだシフト可能である。

【0104】サイクル3の $\phi 1$ 位相期間中は、データはそれより前の全てのステージ内においてシフト可能であるが、データD2をステージFの一次記憶エレメント内にシフトすることは出来ない。ただし、有効なデータが、一旦、二次記憶エレメント内にロードされると、ステージFはこのデータを供給することができない。ステージFはこのイベントを、そのACCEPT信号をLOWの外にセットすることにより送信する。

【0105】ステージFへのACCEPT信号がLOWのままであると仮定すると、Fステージの上流のデータは、次の有効なデータブロックD3がステージEの一次記憶エレメントに到達するまで、ステージの間、及びそれぞれのクロック位相のステージ内においてシフトされることが可能である。図に示すように、サイクル4の $\phi 1$ 位相期間中にこの状態に到達する。

【0106】サイクル5の $\phi 1$ 位相期間中に、データD3のステージEの一次記憶エレメントへのロードが完了する。このデータは更にシフト不可能であるので、ステージEの一次記憶エレメントからのACCEPT信号はLOWにセットされる。上流データは正常にシフト可能である。

【0107】ここで、図2のサイクル5に示すように、パイプラインの下流でに接続されたデバイスがパイプラインデータを受け入れることができると仮定することとする。このデバイスは、サイクル4の $\phi 1$ 位相期間中にACCEPT信号をパイプラインステージF内にHIGHにセットすることによってこのイベントを送信する。ここにおいて、ステージFの一次記憶エレメントはデー

データを右にシフトし、そして、ステージFの一次記憶エレメントは、同様に、新しいデータを受け入れることが可能である。従って、データD1は、サイクル5のφ1位相期間中にシフトされたので、ステージFの一次記憶エレメントは、セーブされなければならないデータを既に含んでいない。従って、サイクル5のφ1位相期間中に、データD2は、ステージF内において、二次記憶エレメントから一次記憶エレメントへシフトされる。ステージFの二次記憶エレメントは、ACCEPT信号をステージEの一次記憶エレメント内にHIGHにセットすることによって新しいデータ及び信号を受け入れることができる。1つのステージ内における、即ち、その二次からその一次記憶エレメントへのデータの転送期間中に、両組の記憶エレメントは同じデータを含むが、二次記憶エレメント内のデータは一次記憶エレメント内に同様に保持されるので、二次記憶エレメント内のこのデータは、データ損失なしに、重ね書き可能である。1つのステージの一次記憶エレメントからその次のステージの二次記憶エレメントへのデータ転送に関しても同様に真である。

【0108】ここにおいて、サイクル5のφ1相期間中、それステージFの一次記憶エレメントへのACCEPT信号がLOWになるものと仮定する。これは、ステージFがパイプラインからデータD2を転送移することができないことを意味する。従って、ステージFは、有効なデータD2の重ね書きを防止するために、その二次記憶エレメントからその一時記憶エレメントへのACCEPT信号をLOWにセットする。ただし、ステージFの二次記憶エレメントに記憶されたデータD2は、損失なしに重ね書きされることは不可能であり、そして、その結果、データD3は、サイクル6のφ0相期間中にステージFの二次記憶エレメント内に転送される。データD4及びD5は、正常状態として下流にシフトされることは出来ない。有効なデータD3が、データD2と共に一度ステージFに記憶されると、ステージFの一次記憶エレメントへのACCEPT信号がLOWである限り、あらゆる二次記憶エレメントは新しいデータを受け入れることが出来ず、そして、二次記憶エレメントは、ステージEへのACCEPT信号をLOWにセットすることによってこれを送信する。

【0109】下流のデバイスからパイプラインへのACCEPT信号がLOWからHIGHへ、あるいは、その逆に変化する場合、この変化は、パイプライン内において、直前を先行する記憶エレメントより更に上流まで伝播してはならない（同一ステージ内か、或いは、先行パイプラインステージ内）。更に、この変化は、当該パイプライン内において、クロック位相につき1つの記憶エレメントブロックだけ上流に伝播する。

【0110】この例に説明されるように、図4、図5、図6、図7に示すパイプライン構造における「1つのス

テージ」のコンセプトは、ある程度知覚に関わる問題である。データがステージの間に所在する場合には、当該データは、ステージ内において（二次記憶エレメントから一次記憶エレメントまで）転送され（上流ステージの一次記憶エレメントから隣接下流ステージの二次記憶エレメントへ）されるので、1つのステージは、図4、図5、図6、図7に示す場合の代わりに、「二次記憶エレメント」によって後続される「一次」記憶エレメントから成ると考えて差し支えない。従って、「一次」および「二次」記憶エレメントのコンセプトは、殆どラベリングの問題である。図4、図5、図6、図7において、データが1つのステージからその次のステージ又はデバイスに転送される場合に、一次記憶エレメントは、それからデータが取り出されるエレメントであり、そして、「二次」記憶エレメントは、同一ステージに対して「入力」記憶エレメントであり得るので、「一次」記憶エレメントは、「出力」記憶エレメントとして参照することも出来る。

【0111】図1、図3に示すように、前述の実施例について説明する場合、ACCEPT及びVALID信号の制御の下でのデータの転送についてのみ言及した。更に、各パイプラインステージの内部記憶エレメント間で供給する以前に、或いは、その次のパイプラインステージに供給する以前に、各パイプラインステージは、それが任意に受け取ったデータも同様に処理することが可能であることを理解されたい。従って、更に図4、図5、図6、図7をも一度参照して、パイプラインステージは入力及び出力記憶エレメントを含み、そして、その記憶エレメントに記憶されたデータを任意に処理するパイプラインの一部として定義できる。

【0112】更に、パイプラインステージFから下流の「デバイス」は、ある種の他のタイプのハードウェア構造である必要はなく、むしろ、他のパイプラインそのもの又はその部分であっても差し支えない。以下に説明するように、パイプラインステージは、下流の記憶エレメントの全てが有効なデータで満たされた場合のみならず、ステージがそのデータの処理を終了するために複数のクロック位相を必要とする場合にも、そのACCEPT信号をLOWにセットすることができる。これは、パイプラインステージがその記憶エレメントの1つ又は両方内に有効なデータを作る場合にも、同様に起こり得る。換言すれば、ステージにとって、直ぐ下流の記憶エレメントが、供給不可能な有効なデータを含むか否かに基づいてACCEPT信号を単に供給することは必要でない。むしろ、ACCEPT信号自体も、隣接記憶エレメント間のデータ用通路を制御するために、当該ステージ内において、或いは、当該ステージにとって外部の回路によって、変更可能である。VALID信号は、類似の方法において同様に処理可能である。

【0113】2線インターフェース（VALID及びA

ACCEPT信号の各々に対してそれぞれ1本のワイヤ)の大きな利点は、パイプラインをその開始ステージまで全経路を後方に向かって伝播するために必要な制御信号なしで、パイプラインを制御する能力を持つことである。図1のサイクル3を再度参照することとし、例えば、ステージFは、データを受け入れることができないことをステージEに「告げ」るが、ステージEはステージDに告げ、そして、ステージDはステージCに告げる。実際問題として、含まれる有効なデータよりも多くのステージが有る場合には、この信号がパイプラインに沿って更に遠くまで後方に伝播する可能性がある。図4、図5、図6、図7に示す実施例、サイクル3において、LOW ACCEPT信号は、ステージEより上流には一切伝播せず、その一次記憶エレメントまでにとどまる。

【0114】以下に説明するように、この実施例は、当該設計を実現するために必要なシリコン部分に特に添付することなしに、この融通性を達成することができる。一般に、データ記憶のために使われるパイプラインにおける各ラッチは、余分にトランジスタ1つだけを必要とする(極めて能率的にシリコンに適合する)。更に、各半ステージにおいてデータラッチと関連するACCEPT及びVALID信号を処理するために、余分に2つのラッチ及び少数のゲートを追加することが好ましい。

【0115】図4、図5、図6、図7に示すようにステージを実現するハードウェア構造を図8に示す。

【0116】説明するための一例に過ぎないが、(オプションとしての組み合わせ論理回路においてそれ以上の操作を行うか、又は、行うことなしに)パイプラインを通して並列に8ビットデータを転送するものと仮定する。ただし、本発明を実現するためには、1ビットデータ又は8ビット未満のデータのいずれかを用いることができることを理解されたい。更に、本実施例に従った2線インターフェースは、あらゆる幅のデータバスと共に使用することに適しており、そして、データバスの幅は、特定アプリケーションの必要性に応じて、ステージ毎に異なることもあり得る。この実施例に従ったインターフェースは、アナログ信号を処理するためにも使用できる。

【0117】既に検討したように、これとは別の従来のタイミング配置を使用可能であるが、インターフェースは、2相非重複クロックによって制御されることが好ましい。図8～図16において、これらのクロック位相信号は、PH0及びPH1として参照される。図8において、各クロック位相信号用に1本のラインを示す。

【0118】入力データは、マルチビットデータバスINデータを介してパイプラインステージに入力され、そして、出力データバスOUTデータを介してその次のパイプラインステージ、或いは、その次の受信回路へ転送される。入力データは、先ず、集散的にLDINとして参照され上記の二次記憶エレメントを構成する一連の

カラッチ(各入力データ信号にたいして1つ)に以下に示す方法によってロードされる。

【0119】図に示す本実施例の例において、全てのラッチのQ出力は、それらのD入力をフォローする、即ち、クロック入力が高レベルである場合、即ち、論理「1」レベルにおいて全てのラッチのQ出力がロードされる。更に、Q出力は、それらの最後の値を保持する。換言すれば、Q出力は、それらの対応するクロック信号の降下落端で「ラッチ」される。

【0120】各ラッチは、そのクロックに対して、1つ又は2つの非重複クロック信号PH0又はPH1(図9、図10に示すように)、又は、これらのクロック信号PH0、PH1のうちの1つ、及び1つの論理信号の理的AND組合わせを持つ。ただし、本発明は、ラッチ動作の適切なタイミングを保証するために従来の方法が適用される限り、クロック信号の上昇端でラッチ作動するラッチ、或いは、他の既知のラッチ用配置を提供することによって同程度に良好に作動する。

【0121】入力データラッチLDINからの出力データは、任意かつオプションとしての組み合わせ論理回路B1を経て供給し、この組み合わせ論理回路は、入力ラッチLDINからの出力データを中間データに変換するために提供されることもあり、次に、この中間データは後で出力データラッチLDOUTにロードされ、この出力データラッチは上記の一次記憶エレメントから成る。

【0122】出力データラッチLDOUTからの出力は、OUTデータとしてその次のデバイスまで下流方向に供給される以前に任意かつオプションとしての組み合わせ論理回路B2を経て同様に供給することが可能である。その次のデバイスとは、他のパイプラインステージ、或いは、パイプラインに接続された他のデバイスであっても差し支えない。

【0123】本発明の実現において、パイプラインの各ステージは、同様に、妥当性検査入力ラッチLVIN、妥当性検査出力ラッチLVOUT、受入れ入力ラッチLA IN、及び受入れ出力ラッチLAOUTを含む。これらの4つのラッチの各々は、簡単な単一ステージラッチであることが好ましい。ラッチU/IN、LVOUT、LA IN、及びLAOUTからの出力は、それぞれ、QVIN、QVOUT、QAIN、QAOUTである。妥当性検査入力ラッチからの出力信号QVINは、入力として直接、或は、中間論理デバイス又は信号を変更する回路を経て妥当性検査出力ラッチLVOUTへ接続される。

【0124】同様に、所定のステージの出力妥当性検査信号QVOUTは、その次のステージの妥当性検査入力ラッチQVINの入力へ、直接、又は、中間デバイス又は論理回路を経て接続しても差し支えない。この場合のデバイスは、妥当性検査信号を変更するデバイスであっても差し支えない。この出力QVINは、同様に、論理

ゲート（以下に説明する）に接続され、この論理ゲートの出力は、受入れ入力ラッチLAINの入力に接続される。受入れ出力ラッチLAOUTからの出力QAOUTは、任意に他の論理ゲートを経て類似の論理ゲート（以下に説明する）に接続される。

【0125】図8に示すように、出力妥当性検査信号QVOUTは、後続するステージによってIN VAL ID信号として受け取られることの出来るOUT VAL IDを形成するか、或いは、単にパイプラインに接続される後続回路に有効なデータを指示する。その次の回路またはステージのデータ受け入れ準備は、信号OUT ACCEPTとして各ステージに指示され、この信号は、好ましくは以下に説明する論理回路を経て、受入れ出力ラッチLAOUTに入力として接続される。同様に、受入れ出力ラッチLAOUTの出力QAOUTは、好ましくは以下に説明する論理回路を経て、受入れ入力ラッチLAINに入力として接続される。

【0126】本発明を実現する際に、妥当性検査ラッチLVINからの出力信号QVIN、QVOUTは、それぞれ、受入れラッチLAIU、LAQUTへの入力を形成するために、受入れ信号QAOUT、OUT ACCEPT1とそれぞれ組合わされる。図8に示す実施例において、これらの入力信号は、それぞれの受入れ出力信号QACUT、OUT ACCEPTの論理的逆として、それぞれの妥当性検査信号QVIN、QVOUTの論理NAND組み合わせとして形成される。従来の論理ゲートNAND1、及びUAND2は、NAND動作を実施し、そして、インバータINV1、INV2は、それぞれの受入れ信号の論理的逆を形成する。デジタル設計技術において周知であるように、NANDゲートの任意又は全ての入力信号が論理「0」状態にある場合、NANDゲートからの出力は論理「1」である。従って、NANDゲートの全ての入力が論理「1」状態にある場合に限り、NANDゲートからの出力は論理「0」である。当該技術分野において同様に周知であるように、例えばINV1のようなデジタルインバータの出力は、その入力信号が「0」である場合に、論理「1」であり、そして、その入力信号が「1」である場合に「0」である。

【0127】従って、「NOT」が2進反転を示す場合には、NANDゲートNAND1への入力は、QVIN、及びNOT (QACUT) である。既知の技術を用いて、受入れラッチLAINへの入力は、次のとおり分解できる。

【0128】

$$\text{NAND}(\text{QVIN}, \text{NOT}(\text{QAOUT})) \\ = \text{NOT}(\text{QVIN}) \text{ OR } \text{QAOUT}$$

換言すれば、信号QVINは「0」であるか、或いは、信号QAOUTが「1」であるか、或は、両方の場合に、インバータINV1、及びNANDゲートNAND

1の組み合わせは論理「1」である。従って、ゲートNAND1及びインバータINV1は、その入力の1つが受入れラッチLAOUTのQAOUT出力に直接結合され、いま1つの入力が妥当性検査入力ラッチLVINの出力信号QVINの反転に結合されている1つの単一ORゲートによって実現可能である。

【0129】同様に、デジタル設計技術において周知であるように、妥当性検査及び受入れラッチとしての使用に適する多くのラッチは、2つの出力Q、及びNOT (Q)、すなわち、Q、及びその論理的逆を持つことができる。従って、この種のラッチを選定した場合には、OPゲート（従って）への1つの入力は、妥当性検査ラッチLVINのNOT (Q) 出力に直接結合されることが可能である。ゲートNAND1及びインバータINV1は、周知の従来の技術を用いて実現することが出来る。ただし、使用されるラッチのアーキテクチャに依じて、出力を反転することのないラッチを使用し、そして、その代りにゲートNAND1、及びインバータINV1を提供することが更に効率的である場合もあり得る。これら両者は、同様に、シリコンデバイスにおいて能率的に実現可能である。従って、Q信号、及び/又は、その論理的逆を生成するために任意の既知の配置を使用することができる。

【0130】双方のクロック信号（出力側におけるPH0、及び入力側におけるPH1）及び同じ側の受入れラッチからの出力が論理「1」である場合、データ、及び妥当性検査ラッチLDIN、LDOUT、LVIN、及びLVOUTは、それらのそれぞれのデータ入力をロードする。従って、クロック信号（入力ラッチLDIN及びLVINに対するPH0）、及びそれぞれの受入れラッチ（この場合のLAIN）の出力は、論理的ANDの方法に使用され、そして、それらが双方共に論理「1」である場合に限り、データはロードされる。

【0131】例えば、ラッチのCMOS実現のような特定のアプリケーションにおいて、ラッチのローディング（図に示すCK、または作動化「入力」を経て）を制御する論理的AND演算は、それぞれの作動化入力信号（例えば、ラッチLVIN、及びLDINのためのPH0、及びQAIN）をラッチの入力ラインに直列接続されたMOSTランジスタへのゲートに接続することにより従来の方法において容易に実現可能である。従って、実際の論理ANDゲートを提供することが必要であり、これによって、高速アプリケーションにおける伝播遅延に起因するタイミングの問題を引き起こす可能性がある。従って、図に示すANDゲートは、様々なラッチの作動化信号を生成するために実施されるべき論理関数を指示するに過ぎない。

【0132】従って、PH0及びQAINが双方共に「1」である場合に限りデータラッチLDINは入力データをロードする。このデータラッチは、これら2つの

信号のどちらかが「0」になる場合に、このデータをラッチする。

【0133】パイプラインステージの入力（および出力）側におけるデータ及び妥当性検査ラッチをクロックするためにはクロック位相信号PH0またはPH1の1つだけが用いられるが、もう一方のクロック位相信号は、同じ側における受入れラッチをクロックするために直接用いられる。換言すれば、パイプラインステージのいずれかの側（入力、または出力）の受入れラッチは、同じ側のデータ及び妥当性検査ラッチを用いて、その「位相外れ」をクロックされることが好ましい。例えば、PH0は、データラッチLDIN及び妥当性検査ラッチLVIN用のクロック信号CKを生成するために使われるが、PH1は、受入れ入力ラッチをクロックするために用いられる。

【0134】2線妥当性検査及び受入れ回路によって増補されたパイプラインの作動例として、当該回路への入力においては、先行パイプラインステージ、或いは、伝送デバイスのどちらからでも、最初には、有効なデータは提供されないものと仮定する。換言すれば、当該システムは極めて最近リセットされたので、図に示すステージへの妥当性検査入力信号IN VALIDはまだ「1」になっていないものと仮定する。更に、システムが最後にリセットされた時から、数回のクロックサイクルが発生し、そして、その結果として、当該回路は定常に到達したものと仮定する。従って、クロックPH0のその次の正の期間中、妥当性検査ラッチLVINからの妥当性検査入力信号QVINは「0」としてロードされる。従って、クロック信号PH1のその次の正の期間中、受入れ入力ラッチLA IN（ゲートNAND1、または他の同等のゲートを介して）への入力は、「1」としてロードされる。すなわち、データ入力ラッチLOINにおけるデータは有効でないので、当該ステージは（当該ステージは一切のデータ価値保管を保持しないので）、当該ステージが入力データの受け入れ準備が整っていることを送信する。

【0135】この例において、データ及び妥当性検査ラッチLOIN及びLVINを作動可能にするために信号IN ACCEPTが用いられることに注意されたい。この時点において、信号IN ACCEPTは「1」であるので、これらのラッチは、従来の透過ラッチとして効果的に作用し、その結果として、INデータバス上のあらゆるデータは、クロック信号PHDが「1」となると直ちに、単に、データラッチLOINにロードされる。勿論、その受入れラッチからの出力QAOUTが「1」である限り、この無効データも、同様に、その次に接続するパイプラインステージのその次のデータラッチLDOUTにロードされる。

【0136】従って、データラッチが有効なデータを含んでいない限り、それぞれのクロック信号のその次の正

の周期中、このラッチは、提供されるあらゆるデータを受け入れる、即ち、「ロード」する。一方、そのような無効のデータは、対応する受入れラッチからの受入れ信号がそれに対してロー（即ち、「0」）であるような一切のステージにロードされない。更に、妥当性検査ラッチへの対応するINVALID（或いは、QVIN）信号がローである限り、妥当性検査ラッチからの出力信号（その次の妥当性検査ラッチへの妥当性検査入力信号を形成する）は「0」のままである。

【0137】データラッチへの入力データが有効である場合には、妥当性検査信号IN VALIDが「1」まで上昇することによってこれを指示する。次に、対応する妥当性検査ラッチの出力は、各クロック位相信号のその次の上昇端において「1」まで上昇する。例えば、クロック位相信号PH0のその次の上昇端において、対応するIN VALID信号がハイになる（すなわち、「1」に上昇する）場合、ラッチLVINの妥当性検査入力信号QVINは「1」に上昇する。

【0138】その代りに、ここで、データ入力ラッチLDINが有効なデータを含んでいるものと仮定する。データ出力ラッチLDOUTの新しいデータを受け入れる準備が整っている場合には、その受入れ信号QAOUTは「1」である。この場合、クロック信号PH1のその次の正の周期中に、データラッチLDOUT及び妥当性検査ラッチVOLTは作動可能化され、そして、データラッチLDOUTは、その入力に所在するデータをロードする。クロック信号は非重複性であるので、上記の動作は、もう一方のクロック信号PH0のその次の上昇端以前に発生する。従って、PH0のその次の上昇端において、先行データラッチ（LDIN）は、データ出力ラッチLOOUTが、ラッチLDINから転送されたデータを安全にラッチし終わるまで、先行ステージからの新しい入力データにおいてラッチしない。

【0139】従って、クロックの交互位相に基づいて作動するので、データを受け取り可能なデータラッチ（ステージ内、或いは、隣接ステージ間）の全ての隣接対によって同じシーケンスが続けられる。未だ供給できない有効なデータを含んでいるために、新しいデータを受け入れる準備が整っていないあらゆるデータラッチは、LOWである出力受入れ信号（その受入れラッチLAからのQA出力）を持ち、そして、そのデータラッチLDIN或いはLDOUTはロードされない。従って、所定のステージ、又は、或るステージの一方の側（入力、または出力）の受入れ信号（受入れラッチからの出力）がLOWである限り、その対応するデータラッチはロードされない。

【0140】好ましい実施例に含まれるリセット機能を図8に示す。この図に示す例において、リセット信号NOTRESET0は、妥当性検査出力ラッチLVOUの反転リセット入力（反転は、従来通り、小さい丸によ

って示す)に接続される。周知のように、これは、リセット信号NOTRESET0が「0」になるときはいつでも、妥当性検査ラッチLVOOUTが「0」を出力するように強制されることを意味する。リセット信号がローになる(「0」になる)場合にラッチをリセットすることの1つの利点は、伝送の遮断によってラッチをリセットすることである。この場合、有効な伝送が始まり、そして、リセット信号がHIGHになる時にはいつでも、これらのラッチは「空の」状態、或いは、リセットされた状態である。従って、リセット信号NOTRESET0はデジタル「オン/オフ」スイッチとして作動する。従って、この信号は、パイプラインを作動化するためにはHIGH値でなければならない。

【0141】パイプラインにおいて有効なデータを保持するラッチの全てをリセットすることが必要ではないことに注意されたい。図8に示すように、妥当性検査入力ラッチLVINは、リセット信号NOTRESET0によって直接リセットされず、間接的にリセットされる。リセット信号NOTRESET0が「0」に降下するものと仮定する。受入れ出力ラッチLAQUT(ゲートNAND1を経て)への入力がHIGHになると、その以前の状態には無関係に、妥当性検査出力信号QVOOUTも同様に「0」まで降下する。同様に、受入れ出力信号QAOUTも「1」に上昇する。次に、このQAOUT値「1」は、「1」として、妥当性検査入力信号QVIN状態には無関係に、受入れ入力ラッチ入力LAINへ転送される。次に、受入れ入力信号QAIは、Nクロック信号PH1の次の立ち上がりエッジにおいて「1」まで上昇する。妥当性検査信号IN VALIDが正しく「0」にリセットされたと仮定すれば、妥当性検査ラッチLVINが直接リセットされていた場合にはたならば、実施された時に、クロック信号PH0の次の立ち上がりエッジにおいて、妥当性検査ラッチLVINからの出力は「C」になる。

【0142】この例が示すように、全ての妥当性検査ラッチをリセットするためには、各ステージ(最終ステージを含む)のただ1つの側においてのみ妥当性検査ラッチをリセットすることだけが必要である。事実、多くのアプリケーションにおいて、他の全ての妥当性検査ラッチをリセットすることは必要でない。即ち、リセット信号NOTRESET0が、クロックの双方の位相PH0、PH1の完全な1サイクルよりも長い期間に亘って所在することが保証可能である場合には、「自動リセット」(リセット信号の後方伝播)が、先行パイプライン

ステージにおける妥当性検査ラッチ期間に対して発生する。実際問題としては、少なくとも所在するパイプラインステージの数と同数のクロックの双方の位相の全サイクル数の期間中に亘りリセット信号が保持される場合、最終パイプラインステージにおける妥当性検査出力ラッチを直接リセットすることだけが必要である。

【0143】図9、図10は、非重複クロック信号PH0、PH1の間の関係、リセット信号の効果、データの保持、及び妥当性検査および受入れ信号の種々異なる順序に対して図8に示す実施例におけるパイプラインステージの図示された2つの側、及びその間へのデータの転送を示す。図9、図10のタイミングダイアグラムに示す例においては、データラッチLDIN、LDOUTからの出力は、介在している論理ブロックB1、B2によるこれ以上の操作なしで供給されるものと仮定されている。これは一例として示すものであり、制約的な意味をもつものではない。あらゆる組み合わせ的論理構造は、連続したパイプラインステージのデータラッチの間、又は、1つの単一パイプラインステージの入力と出力側との間に含ませることが可能であることを理解されたい。入力データに関して実際に図示されている値(例えば、HEXデータワード「aa」、または「04」)も同様に単に図解例に過ぎない。データラッチ又は他の記憶デバイスが、各ビット又は入力ワードの値を収容およびラッチまたは記憶可能である限り、上記のように、入力データバスの幅は任意である(そして、アナログであっても差し支えない)。

好ましいデータ構造 — 「トークン」

図8に示すサンプルアプリケーションにおいて、組み合わせ論理ブロックB1、B2、等々を介して入力データを供給可能にすることからステージを除外する制御回路が無いので、各ステージは、全ての入力データを処理する。更に大きい融通性を提供するために、本発明は、システム全体を通じてデータ及び制御情報を配分するために「トークン」が使われるようなデータ構造を有する。各トークンは、1つまたは複数のトークンワードブロックに分割された一連の2進ビットから成る。更に、ビットは、次に示す3つのタイプのいずれかの1つに相当する、即ち、アドレスビット(A)、データビット(D)、または拡張ビット(B)である。単に例として、必ずしも制約的な意味をもつことなく、データが、1ビット拡張ビットラインを有する0ビットバスを介してワードとして転送されるものと仮定する。4ワードトークンの例を伝送順に次に示す。

第1番目のワード	E	A	A	A	D	D	D	D
第2番目のワード	E	D	D	D	D	D	D	D
第3番目のワード	E	D	D	D	D	D	D	D
第4番目のワード	E	D	D	D	D	D	D	D

拡張ビットEは、各データワードへの追加(好ましくは)として使われることに注意されたい。更にアドレス

フィールドの長さは可変であっても差し支えなく、そして、第1ワードの拡張ビットの直後に伝送されることが

好ましい。

【0144】従って、トークンは、本発明における（2進）デジタルデータの1つ又は複数のワードから成る。これらのワードの各々は、好ましくは並列に順序立てて転送される。ただし、この転送方法は必ずしも必要でなく、既知の技術を用いて直列データ転送も可能である。例えば、動画構文解析系において、制御情報は並列に伝送され、この場合、データは直列に伝送される。

【0145】例として図示するように、各トークンは、好ましくは開始時において、トークンに含まれるデータのタイプを識別する1つのアドレスフィールド（A-ビットのストリング）を持つ。殆どのアプリケーションにおいて、1つの単一ワード、または1つのワードの部分は、アドレスフィールド全体を転送するために十分である。しかし、アドレスフィールド全体を受け取り、そして、デコードするのに十分な長さを持つ部分的なアドレスフィールドのある程度の表現を記憶することができるような対応するパイプラインステージに論理回路が含まれていさえすれば、前記に条件は、本発明にとって必ずしも必要でない。

【0146】アドレスフィールドを伝送するためには専用線または専用レジスタを必要としないことに注意されたい。アドレスフィールドは、データビットを用いて伝送されることに注意されたい。以下に説明するように、特定のアドレスフィールドによって作動化されることが意図されていない場合、即ち、ステージが遅延なしにトークンに沿って供給可能である場合には、パイプラインステージは減速されない。

【0147】アドレスフィールドに続くトークン内の残りのデータは、トークンの使用によって制約されない。これらのDデータビットは、任意の値をとることが可能であり、そして、これらのビットに付けられ意味はここでは重要でない。即ち、データの意味は、例えば、特定の時点に当該データがシステム内のどこに配置されていたかによって、変化することが可能である。アドレスフィールドの後に添付されたデータビットDの数は、必要だけでなく短くても長くても差し支えなく、そして、トークンが異なれば、データワードの数が大幅に変化することがある。アドレスフィールド、及び拡張ビットは、パイプラインステージに対して、制御信号を運ぶために使われる。データフィールド内のワード（Dビットのストリング）の数は随意で差し支えないので、データフィールドにおいて運ばれる情報も、それに応じて変化可能である。従って、以下の説明は、アドレス及び拡張ビットの使用に向けられる。本発明において、回路内の多数のブロックが、比較的簡単な構成にまとめて接続されている場合、トークンは特に有用なデータ構造である。最もシンプルなコンフィギュレーションは、例えば、その1つが図1に示されているような、処理ステップのパイプラインである。ただし、トークンの使用は、パイプライン

構造における使用にのみ制約されない。

【0148】各ボックスは完全なパイプラインステージを表すものと再度仮定する。図1のパイプラインにおいて、データは図の左から右に流れる。データは、マシンに入り、そして、処理ステージAに供給する。このステージはデータを修正する場合もあれば、しない場合もあり、そして、データをステージBへ供給する。修正される場合には、複雑であることもあり、そして、一般に、任意のステージに流入するデータアイテムの数は、流出数と同じでない。ステージBは、再びデータを修正し、そして、それをステージCに供給する、等々。この種の設計の場合には、データにとって、反対方向に流れることは不可能である。従って、例えば、ステージCはデータをステージAに供給できない。この制約条件は、完全に受け入れられる場合が多い。

【0149】一方、これら2つのブロックの間が直接接続されていない場合であっても、ステージAが情報をステージCに伝達することができることは非常に望ましい。ステージAとCの通信はステージBをへる場合に限られる。トークンの1つの利点は、トークンがこの種類の通信を達成する能力を備えていることである。トークンを認識しないあらゆる処理ステージは変更しないトークンをその次のブロックに単に供給するに過ぎない。

【0150】この例によれば、1つの拡張ビットは、各トークンにおけるアドレス及びデータフィールドと共に伝送されるので、処理ステージは、そのアドレスを復号化しなければならないことを一切必要とせず、トークン（随意の長さであり得る）を供給することができる。この例によれば、HIGH（「1」）である拡張ビットをその中に含むあらゆるトークンには、同じトークンの一部である後続ワードによって後続される。この後続ワードも、同様に、拡張ビットを持ち、このビットは、トークン内に更に別のトークンワードがあるかどうかを示す。ステージがその拡張ビットがLOW（1つの「0」）であるようなトークンワードに遭遇した場合には、そのトークンワードが当該トークンの最後ワードであることが分かる。従って、その次のワードは、新しいトークンの第1のワードであると仮定される。

【0151】処理ステージの簡単なパイプラインは特に有用であるが、トークンは、更に複雑なコンフィギュレーションの処理エレメントに適用可能であることが分かることに注意されたい。更に複雑な処理エレメントの例を次に示す。

【0152】本発明の場合には、「0」にセットされた拡張ビットを与えることにより所定のトークンの最後のワードを送信するために、拡張ビットの状態を使用する必要はない。好ましい実現計画への1つの代替案は、トークンの最後のワードの代りにトークンの第1のワードを示すように拡張ビットを移動させることである。これは、符号復号化ハードウェアを適切に改造することによ

って達成できる。

【0153】トークンの第1のワードでなくて最後のワードを送信するために本発明の拡張ビットを用いた場合の利点は、トークンが拡張ビットを持っているかどうかに応じて、回路のブロックの作動様式を修正することが有用である場合が多い。前記の論旨の例は、量子化表（一般にはメモリーデバイス）に記憶された動画量子化値を処理するステージを作動化するトークンである。例えば、64の8ビット任意2進整数を含む表である。

【0154】新しい量子化表をパイプラインの量子化器ステージにロードするためには、「QUANT・テーブル」トークンが量子化器に送られる。そのような場合、例えば、トークンは、65のトークンワードから成る。第1のワードは、コード「QUANT・テーブル」を含む、即ち、量子化表を作る。これには、量子化表の整数である64のワードが後続する。

【0155】動画データをコード化する場合、この種の量子化表を伝送することが時おり必要である。この機能を達成するために、拡張ワードなしのQUANT・テーブルトークンを量子化器ステージに送ることが出来る。このトークンを見て、そして、その第1のワードの拡張ビットがLOWであることに気が付いた場合、量子化器ステージは、その量子化表を読みとり、そして、64の量子化表値を含むQUANT・テーブルトークンを作成することが出来る。第64番目の量子化表の値に相当するLOW拡張ビットによって当該トークンの新規な終端が示されるまで、当該拡張ビットがHIGHであり、そして、HIGH拡張ビットの状態で当該トークンが継続するように、第1のワード（LOWであった）の拡張ビットが変更される。これは、システムを通して一般的な方法で進行し、そして、ビットの流れの中にコード化される。

【0156】この例を用いて説明を続けることとし、量子化器は、QUANT・テーブルトークンの第1ワードがその拡張ビットをセット済みであるか否かに応じて、新規な量子化表をそれ自体のメモリーデバイスにロードするか、或いは、その表を読みとることのどちらかが可能である。

【0157】拡張ビットを使用して1つのトークン内の第1のトークンワードか或は最後のトークンワードのどちらを送信するかを選定は、その中でパイプラインが使用されるシステムに依存する。両代替案共に、本発明に従って可能である。

【0158】好ましい拡張ビット案の別の代替案は、トークンの開始に際して長さカウントを含むことである。例えば、トークンが非常に長い場合、この種の装置が効率的であることもある。例えば、所定のアプリケーションにおける一般的なトークンの長さは1000ワードであるものと仮定する。図に示す拡張ビット案（各トークンワードに付されたビットを持つ）を用いた場合、ト

クンは、全ての拡張ビットを含むために、1000の付加的ビットが必要である。ただし、2進フォームにおいてトークンの長さをコード化するためには、僅かに10ビットが必要である。

【0159】従って、長いトークンの用途はあるが、経験によれば、短いトークンの用途も多いことが判明している。ここでは、好ましい拡張ビット案が有利である。トークンの長さわずかの1ワードである場合には、これを送信するために、わずか1ビットが必要とされる。ただし、カウント案には、前記の場合と同じ10ビットを必要とする。

【0160】長さカウント案の欠点を次に示す。1) 短いトークンに対して非能率的である。2) トークンに対して最大の長さの点で制約する（1023ワード未満であれば、僅かに10ビットでカウント可能である）3) カウントを生成する（おそらくトークン開始時ある）以前にトークンの長さが既知でなければならない。4) トークンを扱う全ての回路ブロックは、ワードをカウントするためのハードウェアを備える必要がある。5) カウントが腐敗した場合（データ伝送エラーのために）、回復が達成可能かどうか不明瞭でない。

【0161】本発明に基づく拡張ビット案の利点を次に示す。1) 未認識トークンは、拡張ビットのみを考察することによって正しく供給出来るので、パイプラインステージは、全てのトークンを復号化する回路ブロックを備える必要がない。2) 拡張ビットの符号化は、全てのトークンに関して同じである。3) トークンの長さに関して限界がない。4) この計画案は、短いトークンに対して効率的である（トークンの長さを表すためのオーバーヘッドに関して）。5) エラー回復が自然に達成される。拡張ビットが腐敗した場合には、1つのランダムトークンが生成されるか（「1」から「0」までの腐敗した拡張ビットに関して）、或いは、トークンは失われる（腐食拡張ビットは「0」から「1」まで）。更に、問題は関係するトークンに限って局部的に所在する。当該トークンの後で、正しい動作が自動的に再開される。

【0162】更に、アドレスフィールドの長さが変化しても差し支えない。これによって、最も一般的なトークンを非常に少ない数のワードに圧搾することが可能であるので、高度に有利である。これは、全ての処理ステージが全バンド幅で連続的に作動可能であることを保証するので、結果的に、動画データパイプラインシステムにおいて非常に重要である。

【0163】本発明に基づき、可変長さのアドレスフィールドを可能にするために、ランダムデータが後続する短いアドレスが、長めのアドレスと決して混同されないようにアドレスが選定される。アドレスフィールドをコード化するための好ましい技術（意図したパイプラインステージを作動化するための「コード」としても役立つ）は、ハフマンによって最初に開示され通称「ハフマ

ンコード」と呼ばれる周知の技術である。ただし、当該技術分野における通常の熟練者であれば、他の符号化案を用いても成功するであろうと言うことは理解される筈である。

【0164】ハフマン符号化はデジタル設計の分野においてよく理解されているが、一般的な背景を提供するために次の例を示す。

【0165】ハフマンコードは、記号列によって構成されたワードから成る（例えば本発明のようなデジタルシステムの文脈においては、記号は、一般に2進数字である）。コードワードの長さは可変せあり、そして、ハフマンコードワードの特殊な性質は、短めのコードワードを形成する記号で始まる長めのコードワードは有り得ないようにコードワードが選択されることである。本発明に基づけば、トークンアドレスフィールドは、既知のハフマン符号化技術を用いて選定されることが（必ずしも必要ではないが）好ましい。

【0166】更に、本発明においては、第1のワードトークンの最上位ビット（MSB）においてアドレスフィールドが始まることが好ましい。（MSBに関する明示は任意であり、そして、本案は、MSBに関する様々な明示を収容するように修正可能である。）アドレスフィールドは、より重要度の低い隣接ビットを介して継続する。特定のアプリケーションにおいて、トークンアドレスが複数のトークンワードを必要とする場合には、あらゆる所定のワード内の最下位のビットであるアドレスフィールドは、その次のワードの最上位ビットにおいて継続する。アドレスフィールドの最小の長さは1ビットである。

【0167】本発明において使用されるトークンを生成するためには、数種の既知ハードウェア構造のうちの任意の1つを使用することが出来る。この種の構造の1つは、マイクロプログラムされた状態マシンである。ただし、既知のマイクロプロセッサ、または他のデバイスを使用しても差し支えない。

【0168】本発明に基づくトークン計画案の主要な利点は、予測されなかった必要性への適応性を備えていることである。例えば、新規なトークンが導入される場合、この導入は極く少数のパイプラインステージにのみ影響するのが一般的である。最も有り得る場合は、僅かに2つのステージ、または回路ブロックが影響されることである。即ち、一方のブロックはまず第一にトークンを生成し、そして、いま一方のブロック又はステージは、この新しいトークンを扱うために新規に設計、或いは、修正されたものである。その他のパイプラインステージは、一切改造する必要のないことに注意された。い。それどころか、これらのステージは、この新規なトークンを認識せず、従って、改造されないトークンを供給するので、これらのステージの設計を改造することなしに、新規なトークンを扱うことができる。

【0169】設計済みの既存のデバイスを、実質的に影響されない状態にしておく能力を持つことは本発明の明瞭な利点である。1つのチップ・セット内の幾つかの半導体チップを、前記のチップ・セット内の前記以外の幾つかの半導体チップを設計面において改良することにより、完全に影響を受けない状態にしておくことが可能である。これは、顧客およびチップメーカー両方の立場から有利である。改造が、設計変更によって全てのチップが影響されることを意味するとしても（集積化の進歩レベルが徐々に高くなり、システム内におけるチップの個数が低下するような条件）、同じ設計を再使用できるので、市販されるまでに要する時間の点において、達成可能な程度よりも優れていると言うかなりの利点が残される。

【0170】特に、2ワードアドレスを含むようにトークンセットを拡張することが必要となった場合に起きる条件に注意されたい。この場合においてさえも、既存の設計を改造することは未だ必要でない。パイプラインステージにおけるトークンデコードは、この種トークンの第1ワードを復号化を試み、そして、第1ワードがトークンを認識しないことを結論する。次に、このトークンデコードは、この動作を正しく実施するために拡張ビットを用いることによって改造されたトークンを供給する。このトークンデコードは、第2ワードは、このトークンデコードが認識しないトークンのデータフィールドの一部であると「仮定する」ので、（当該トークンがアドレスビットを含むとしても）このトークンデコードは、当該トークンの第2のワードを復号化しようと試行しない。

【0171】多くの場合、パイプラインステージ、または接続された回路ブロックはトークンを改造する。これは、一般に、ただし、必要条件としてではなく、トークンデータフィールドを改造する形をとる。更に、特定のデータワードを除去するか、或いは、新規なデータワードを付加するかいずれかの方法により、改造しようとするトークン内のデータワードの個数を変更するのが一般的である。場合によっては、トークンは、トークンの流れから完全に除去される。

【0172】大抵のアプリケーションにおいて、パイプラインステージは、一般に、2、3のトークンを復号化する（トークンによって作動化される）に過ぎない。即ち、ステージは、他のトークンを認識せず、そして、他のトークンを変更しないで供給する。非常に多くの場合、ただ1つのトークン、即ち、データトークンアード自体が復号化される。

【0173】多くのアプリケーションにおいて、特別のステージのオペレーションは、自体の過去のオペレーションの結果に依存する。従って、ステージの「状態」は、その前の状態に依存する。換言すれば、ステージは、記憶された状態情報に依存し、このことは、当該ス

テージが、1つ又はそれ以上前のクロックサイクルにおけるそれ自身のヒストリに関するいくつかの情報を保持しなければならないことを述べる他の方法である。本発明は、この種の「状態マシン」ステージを含むパイプラインにおける使用、並びに、データバスにおけるラッチが簡単なパイプラインラッチであるようなアプリケーションにおける使用に良く適する。

【0174】本発明に基づく2線インターフェースが、この種「状態マシン」(state machine)回路に対して適合性を有することは、発明の重要な利点である。これは、データバスがステートマシンによって制御されつつある場合に特に真である。この場合、上記の2線インターフェース技術は、マシンの「現行状態」がパイプラインにおいて制御中のデータと歩調を合わせて留まっていることを保証するために使用できる。

【0175】トークンアドレスフィールドを復号化するためにパイプラインステージに含まれる回路の1つの例の簡素化された構成図を図11に示す。この図は、「状態マシン」の特性を持つパイプラインステージを示す。トークンの各ワードは1つの「拡張ビット」を含み、このビットは、当該トークン内にこれより多くのワードがあればHIGHであり、或いは、これがトークンの最終ワードであるばLOWである。これがトークンの最終ワードである場合には、その次の有効なデータワードは、新規なトークンの開始であり、従って、そのアドレスは復号化されなければならない。従って、あらゆる所定のワードにおけるトークンアドレスを復号化するかどうかに関する決定は、前の拡張ビットの値を知ることに依存する。

【0176】図面を簡潔にする目的だけのために、2線インターフェース(受入れ、及び妥当性検査信号、及びラッチと共に)は図示しないこととし、そして、回路をリセットすることを扱う全ての詳細については省略することとする。以前と同様に、8ビットデータワードは、制約のためでなく、単なる例として仮定される。

【0177】この一例としてのパイプラインステージは、データビット及び拡張ビットを1つのパイプラインステージだけ遅延させる。更に、このステージは、データトークンを復号化する。回路出力にデータトークンの第1ワードが供給される時点において、信号「データADDR」が作られ、そして、HIGHにセットされる。データビットは、ラッチLDIN及びLDOUTによって遅延され、これらラッチの各々は、この例(8入力8出力ラッチに対応する)に使われる8データビットに対して8回繰り返される。同様に、拡張ビットは、拡張ビットラッチLEIN、及びLEOUTによって遅延される。

【0178】この例において、ラッチLEPREVは、拡張ビットの最も最近の状態を記憶するために備えられる。拡張ビットの値はLEINにロードされ、そして、

非重複クロック位相信号PH1のその次の立ち上がりエッジにおいてLEOUTにロードされる。従って、ラッチLEOUTは、現行拡張ビットの値を含むが、非重複2相クロックの後半期間中のみに限られる。ただし、ラッチLEPREVは、この拡張ビット値をクロック信号PH0、即ち、拡張ビット入力ラッチLEINを作動可能にする同一信号のその次の立ち上がりエッジにおいてロードする。従って、ラッチLEPREVの出力QEPREVは、前のPH0クロック位相の期間中、拡張ビットの値を保持する。反転Q出力からのデータワード出力の5ビットにラッチLDINの非反転MD[2]を加えた結果は、一連の論理ゲートNAND1、NAND2、及びNOR1における前の拡張ビット値QEPREVと組合わされる。これらのオペレーションについては、デジタル設計技術の分野において周知である。称呼「NMD[m]」は、ミッドデータワードMD[7:0]のビットmの論理的な反転を示す。既知のブール代数技術を用いると、前の拡張ビットが「0」(QPREV=「0」)である場合に限りこの論理ブロック(NOR1からの出力)からの出力信号SAはHIGH(「1」)であり、ということが示され得るそして、非反転Qラッチ(オリジナルの入力ワード)LDINの出力におけるデータワード構造は「000001xx」であることが分かる。即ち、5つの高位ビットであるMD[7]-MD[3]ビットは全て「0」であり、そして、ビットMD[2]は「1」であり、そして、ゼロ-ワン位置におけるビットはあらゆる任意の値を持つ。

【0179】従って、4つの可能なデータワードは(「xx」の4つの順列がある)、SA、及びひいては、入力SAがその入力に接続されているアドレス信号ラッチLADDRの出力をHIGHにする。換言すれば、このステージは、4つの可能な適正トークンの1つが供給される場合、及び前の拡張ビットがゼロであった場合、すなわち、前のデータワードが、前の一連のトークンワードにおける最終ワードであって、現行トークンワードが現行トークンにおける第1トークンワードであることを意味する場合に限り、作動化信号(データADDP「1」)を供給する。

【0180】従って、ラッチLEPREVからの信号QPREVがLOWである場合、ラッチLDINの出力における値は、新規なトークンの第1ワードである。ゲートNAND1、NAND2、及びNOR1は、データトークン(000001xx)を復号化する。ただし、このアドレス復号化信号SAは、信号データADDRが出力データOUTデータ、及びOUTEXTNと同じタイミングを持つように、ラッチLADOPにおいて遅延される。

【0181】本発明に基づく状態従属パイプラインステージの別の簡単な例を図12に示す。この例は、前の出力拡張ビットOUTEXTNの値を示すために信号LA

10

20

30

40

50

STOUTEXTNを生成する。それぞれ現行および最終拡張ビットラッチLEOUT及びLEPEVへの2つの作動可能化信号(CK入力における)のうちの1つの信号は、データが有効なであり、そして、受け入れられつつある場合に(出力妥当性検査および受入れラッチLVOOUT及びLAOUTからのQ出力はそれぞれHIGHである)、これらのラッチがこれらのラッチに対する新規な値だけをロードするように、ゲートAND1から派生する。このようにして、これらの信号は有効な拡張ビットのみを保持し、そして、有効でないデータと関連している真でない値はこれらの信号にはロードされない。図12に示す実施例において、2線有効/受入れ可能ロジックは、下流の受入れ信号、及び妥当性検査ラッチLVIN及びLVOOUTそれぞれの非反転出力から成る入力信号を供給されるOR1及びOR2ゲートを含む。これは、ラッチが反転出力を持つ場合に、図8におけるゲートNAND1/2、及びINV1/2が置き換え可能であるような1つの方法を示す。

【0182】これは、「状態従属」パイプラインステージの非常に簡単な例、即ち、ただ1つの単一ビットの状態に依存する例であるが、パイプラインステージ間においてデータが実際に転送される場合に限り、状態情報を保持する全てのラッチが更新されることは全体的に真である。換言すれば、データが双方共有効であり、そして、その次のステージによって受け入れられつつある場合に限られる。従って、この種のラッチが適切にリセットされていることを保証するために、注意しなければならない。

【0183】本発明に基づいてトークンを生成および使用することは、パイプラインを介してデータを転送するための周知の符号化技術よりも数々の利点を提供する。

【0184】第1に、既に説明したように、一般のトークンの効率的な表現を提供するために、トークンのアドレスフィールドの長さは可変であっても差し支えない(そして、例えば、ハフマン符号化を利用できる)。

【0185】第2に、トークンの長さについて一貫性のある符号化を行うことにより、所定のパイプラインステージにおけるトークンデコード回路により当該トークンが認識されない場合であっても、トークンの終了が(従って、その次のトークンのスタートが)正しく処理されることを可能にする(簡単で非操作的な転送を含む)。

【0186】第3に、未認識トークンを扱うための(即ち、変更されないままでトークンを供給するための)規則及びハードウェア構造により1つのステージとパイプライン内の最も近い隣接ステージでない下流のステージとの間の通信を可能にする。更に、これは、既存のパイプラインステージの大規模な再設計を必要とすることなくトークンセットの将来における変更を可能にするので、パイプラインの拡張可能性および効率的適応性を増大する。本発明のトークンは、以上および以下において

説明されるように、2線インターフェースと共に使用した場合に、特に有用である。

【0187】図13及び図14に示す例としてのパイプラインステージの機能について次に説明する。ステージが予定されたトークン(この例においてデータトークンと称する)を処理しつつある場合に、当該ステージは、このトークンにおけるデータトークンのアドレスフィールドを含む第1ワード以外の全てのワードを二重化する。一方、ステージが、他の種類の任意のトークンを処理しつつある場合、当該ステージは、全てのワードを削除する。全体的な効果は、出力において、データトークンのみが現れ、そして、これらのトークン内の各ワードは2度繰り返される。

【0188】ここに図示されたシステムの多数の部品は、図4、図5、図6、及び図7に示す遙かに簡単な構造に示されている部品と同じであっても差し支えない。このことは重要な利点を例証する。僅かに改造するか、或は、一切の改造なしに、同じ2線インターフェースを使用できるので、更に複雑なパイプラインステージであっても、融通性および弾力性に関して同じ利点を有する。

【0189】図13、図14に示すデータ二重化ステージは、所定のアプリケーションにおいてパイプラインステージが遂行することのできる無数にある異なるタイプのオペレーションのほんの一例に過ぎない。ただし、この「二重化ステージ」は、「ボトルネック」を形成する可能性があり、その結果として、この実施例に基づくパイプラインが「一緒にバックとなる」ようなステージを示す。

【0190】「ボトルネック」は、そのオペレーションを遂行するために比較的長い時間を要するか、或いは、パイプラインにおいて、受け取ったよりも、より多くのデータを作るあらゆるステージであり得る。更に、この例は、この実施例に基づいた2線受け入れ/有効インターフェースが異なるアプリケーションに非常に容易に適応可能であることを示す。

【0191】図13、図14に示す二重化ステージは、図11の例に示すように各々ステージの入力および出力における拡張ビットの状態をラッチする2つのラッチLEIN及びLEOUTを持つ。図13、図14に示すように、入力拡張ラッチLEINは、入力データラッチLDIN及び妥当性検査信号INVALIDと共に同期的にクロックされる。参照を容易にするために、二重化ステージに含まれる種々のラッチは、それぞれ特有の出力信号と共に対を構成する。

【0192】二重化ステージにおいて、データラッチLDINからの出力は、MIDデータと称する中間データを形成する。この中間データワードは、中間受入れ信号(図13、図14において「MID ACCEPT」と参照)がHIGHにセットされている場合に限りデータ

出力ラッチLDOUTにロードされる。

【0193】図13、図14に示す受入れラッチLA IN、LAOUTの下回路部分は、データを二重化するために使われる様々な内部制御信号を生成するために、基礎的なパイプライン構造に付加される回路である。これらの信号には、当該回路が有効なデータトークンを処理中であることを示す「データトークン」信号、及びデータの二重化を制御するために使われるNOT DUPLICATE信号が含まれる。回路がデータトークンを処理中である場合には、NOT DUPLICATE信号はHIGH状態とLOW状態の間でトグルし、そして、これによって、当該トークン内の各ワードを1度（1度に限る）二重化させる。回路が有効なデータトークンを処理中である場合には、NOT DUPLICATE信号はHIGH状態に保持される。従って、これは、処理されつつあるトークンワードが二重化されないことを意味する。

【0194】図13、図14に示すように、8ビット中間データワードの上位6ビット、及びラッチLI1からの出力信号QI1は、フォーム入力から論理ゲートNOR1、NOR2、NAND18のグループへの入力を形成する。ゲートNAND18からの出力信号はS1と表示される。周知のブール代数を用いて、出力信号QI1が「1」であり、そして、MIDデータワードが次に示す構造を持つ間合いに限り信号S1が「0」とあることを示すことが出来る。「000001xx」即ち、上位5ビットは全て「0」であり、ビットMIDデータ[2]は「1」であり、そして、MIDデータ[1]におけるビット及びMIDデータ[0]位置はあらゆる任意の値を持つ。従って、信号S1は、MIDデータ信号が予定された構造を持ち、そして、ラッチLI1からの出力が「1」である場合に限りローである「トークン識別信号」として作用する。ラッチLI1、及びその出力QI1の性質について以下に更に説明する。

【0195】ラッチLO1は、中間拡張ビットの最後の値をラッチする機能を遂行し（「MID EXTN」及び信号S4として表示）、そして、この値をクロック位相PH0のその次の立ち上がりエッジにおいてラッチLI1にロードする。このラッチの出力はビットQI1であり、そして、信号S1を形成するトークン復号化論理グループへの入力の1つである。既に説明したように、信号S1は、信号QI1が「1」である場合に「0」までただ降下する（そして、MIDデータ信号は予定された構造を持つ）。従って、信号S1は、最後の拡張ビットが「0」であった場合には何時でも「0」までただ降下し、前のトークンが終了したことを示す。従って、MIDデータワードは、新しいトークンにおける第1データワードである。

【0196】ラッチLO2、及びLI2は、NANDゲートNAND20、及びNAND22と共に、信号デー

タトークンのための記憶装置を形成する。正常な条件において、NAND20への入力における信号QI1、及びNAND22への入力における信号S1は、両方共論理「1」である。再度ブール代数の技術により、この条件において、これらのNANDゲートがインバータと同じ方法において作動することがわかる。すなわち、ラッチLI2の出力からの信号QI2は、NAND20において反転され、そして、次に、この信号は、NAND22によって再び反転されて信号S2を形成する。この場合、この通路には2つの論理的な反転があるので、信号S2は、QI2と同じ値を持つ。

【0197】どのように、ラッチLO2の出力における信号データトークンがLI2への入力を形成するということが分かる。その結果、QH1及びS1は両方共HIGHであるという条件に留まる限り、信号データトークンは、その状態（「0」、または「1」）を保持する。クロック信号PH0、及びPH1がラッチ（それぞれLI2、及びLO2）をクロックしつつある場合にも、これは真である。信号QI1、及びS1の1つ、または双方共に「0」である場合、データトークンの値は変化可能であるに過ぎない。

【0198】既に早期に説明したように、前の拡張ビットが「0」であった場合に信号QI1は「0」である。従って、MIDデータ値がトークンの第1ワードである場合には何時でも、前期の信号は「0」である（そして、当該トークン用のアドレスフィールドが含まれる）。この条件において、信号S1は、「0」または「1」のいずれかである。既に早期に説明したように、MIDデータワードがこの例においてはデータトークンを示す予定された構造を持つ場合には、信号S1は「0」である。MIDデータワードが他の構造を持つ場合には（当該トークンがデータトークンでない他のトークンであることを示す）、S1は「1」である。

【0199】QI1が「0」、及びS1が「1」である場合には、これは、データトークン以外のトークンが在ることを示す。デジタル電子工学の分野において周知されているように、NAND20の出力は「1」である。NANDゲートNAND22はこれを反転し（既に説明したように）、そして、信号S2は「0」である。その結果、この「0」値は、その次のPH1クロック位相の開始に際してラッチLO2にロードされ、そして、データトークン信号は「0」になり、当該回路がデータトークンの処理中でないことを示す。

【0200】QI1が「0」であり、そして、S0が「0」であり、それによりデータトークンを示す場合には、（NAND20の出力からのNAND22への他の入力であるにも拘わらず）信号S2は「1」である。その結果、この「1」値は、その次のPH1クロック位相の開始に際してラッチLO2にロードされ、そして、データトークン信号は「1」になり、当該回路がデータ

ークンを処理中であることを示す。

【0201】NOT DUPLICATE信号（出力信号QO3）は、クロックPH0のその次の立ち上がりエッジにおいてラッチL13にロードされる。ラッチL13からの出力信号QI3は、信号S3を形成するために、ゲートNAND24における出力信号Q12と組合わされる。以前同様ブール代数を用いて、信号QI2及びQI3両方の値「1」である場合に限り、信号S3が「0」であることが分かる。信号QI2が「0」になる場合、即ち、データトークン信号が「0」である場合、10 信号S3は「1」になる。換言すれば、有効なデータトークン（QI2=0）がない場合、或いは、データワードが二重化（QI3=0）ではない場合、信号S3はハイになる。

【0202】ここで、データトークン信号が、1クロック信号以上に亘ってHIGHのままであるものと仮定する。NOT DUPLICATE信号（QO3）はラッチL13に「フィードバック」され、そして、ゲートNAND24によって反転される（その別の入力QI2はHIGHに保持される）ので、出力信号QO3は「0」20 と「1」との間でトグルする。ただし、有効なデータトークンが無い場合には、信号QI2は「0」であり、そして、DATEトークン信号がもう一度「1」になるまで信号S3及び出力QO3はHIGHに強制される。出力QO3（NOT DUPLICATE信号）は同様にフィードバックされ、そして、信号QA1及びQO3の両方の値が「1」である場合に限りそれらの出力が

「1」であるような一連の論理ゲート（一緒にANDゲートを形成するNAND16及びINV16）における受入れラッチLA1Nからの出力QA1と組合わされ30 る。図13、図14に示すように、ANDゲート（ゲートINV16にによって後続されるゲートNAND16）からの出力も同様に受入れ信号IN ACCEPTを形成する。この受入れ信号は、すでに述べたように、2線インターフェース構造において用いられる。

【0203】受入れ信号IN ACCEPTは、ラッチLDIN、LEIN、及びLVINに対する作動可能化信号として用いられる。その結果、NOT DUPLICATE信号がローである場合には、受入れ信号IN ACCEPTもローであり、そして、これら3つのラッチ全てが無能化され、そして、それらの出力に記憶された値を保持する。NOT DUPLICATE信号がHIGHになる時まで、ステージは、新しいデータを受け入れない。これは、受入れラッチLA1Nからの出力をハイに強制するために、上記の必要条件に追加される。

【0204】有効なデータトークンが在る（データトークン信号QO2が「1」である）限り、信号QO3は、HIGH状態とLOW状態の間でトグルする。その結果、入力ラッチが作動化され、そして、長くとも、両方のクロック位相PH0、PH1の1つおきの完全サイク50

ルの期間に亘ってデータ受け入れ可能となる。「HIGH」OUT ACCEPT信号によって指示されるように、その次のステージのデータ受け入れ準備が整うことと言う追加的条件は、勿論、依然として満足されていない。従って、少なくとも2つの全クロックサイクルの期間に亘って、出力ラッチLDOUTは、同じデータワードを出力バスOUTデータに配置する。有効なデータトークン（QO2 HIGH）が所在し、そして、妥当性検査信号QVOUTがHIGHである場合に限りOUT VALID信号は「1」である。

【0205】MIDデータに対応する拡張ビットである信号QEINは、信号S4を形成するために、一連の論理ゲート（INV10、及びNAND10）において信号S3と組合わされる。データトークンの表示期間中、各データワードMIDデータは、それを出力ラッチLDOUTに2度ロードすることによって反復される。これらの第1の期間中、S4は、NAND10の作用によって「1」に強制される。信号S4は、OUTデータ

【7:0】を形成するためにMIDデータがLDOUTにロードされるのと同時にOUTEXTNを形成するために、ラッチLEOUTにロードされる。

【0206】このように、最初には所定のMIDデータがLEOUTにロードされ、関連しているOUTEXTNがハイに強制されるが、2番目には、OUTEXTNは信号QEINと同じである。ここで、QEINがローであることが既知である1つのトークンの最終ワードの期間中における条件について考察することとする。第一の期間中、MIDデータがLDOUTにロードされ、OUTEXTNは「1」であり、そして、第2の期間中、OUTEXTNは「0」であって、当該トークンの真の終了を示す。

【0207】妥当性検査ラッチLVINからの出力信号QVINは、信号S5を形成するために、同様のゲート組合わせ（INV12、及びNAND12）において信号QI3と組合わされる。周知のブール技術を用いることにより、妥当性検査信号QVINがHIGHである場合、或いは、信号QI3がローである場合（データが二重化であることを示す）のいずれかの場合において、信号S5がHIGHであることがわかる。MIDデータがLDOUTにロードされ、そして、中間拡張ビット（信号S4）がLEOUTにロードされると同時に、信号S5が妥当性検査出力ラッチLVOUTにロードされる。同様に、信号S5は、出力妥当性検査信号OUT VALIDを形成するために、論理ゲートNAND30、及びINV30における信号QO2（データトークン信号）と組合わされる。既に早期に述べたように、有効なトークンが在り、そして、妥当性検査信号QVOUTがハイである場合に限り、OUT VALIDはHIGHである。

【0208】本発明において、MID ACCEPT信

号は、ラッチLO1、LO2、及びLO3に対して2つの作動可能化信号のうちの1つとして用いられる信号S6を形成するために周知のAND機能を遂行する一連の論理ゲート(NAND26、及びINV26)における信号S5と結合される。MID ACCEPT信号がHIGHであり、そして、妥当性検査信号QVINがハイであるか、或いは、トークンが二重化であるか(QI3は「0」である)いずれかの場合に信号S6は「1」に上昇する。信号MID ACCEPTがHIGHである場合には、ステージの入力において有効な入力データがロードされる時にはいつでも、クロック信号PH1がハイである場合、或いは、ラッチされたデータが二重化である場合、ラッチLO1-LO3は作動可能化される。

【0209】以上の検討から、図13、図14に示すステージが、妥当性検査および受入れ信号の制御の下にステージ間のデータを受け取り、そして、転送することが分かる。ただし、前の実施例の場合のように、入力側における受入れラッチLAINからの出力信号がトグルする二重化信号と結合されることは例外であり、その結果、データワードは、新規なワードが受け入れられる以前に2度出力される。

【0210】例えばNAND16及びINV16のような各種論理ゲートは、勿論、等価論理回路(この場合には、1つの単一ANDゲート)によって置き換えることが可能である。同様に、例えば、ラッチLEIN、及びLVINの出力が反転されつつある場合には、インバータINV10、及びINV12は必要でない。その代わりに、ゲートNAND10、及びNAND12への対応する入力、これらのラッチの反転している出力に直接結合することが可能である。適切な論理演算が遂行される限り、ステージは、同じ方法において動作する。データワード、及び拡張ビットは、依然として二重化される。

【0211】トークンの第1データワードが当該ワードの第3位置に「1」を配置し、そして、5つの高位ビットに「0」を配置するまでは、図に示すステージが遂行する二重化機能は実施されないことに注意されい。(勿論、他の論理ゲートおよび図に示すNOR1、NOR2、NND18ゲート以外の接続を選定することにより、必要とされるパターンは容易に変更および設定可能である。)

更に、図13、図14に示すように、第1データワードが記述の構造を持たない限り、OUT VALID信号は、トークン全体の期間中に互って、ローに強制される。これは、二重化プロセスを起こさせる1つのトークンを除く全てのトークンがトークンの流れから削除されるような効果を発揮する。その理由は、出力端子(OUT DATA、OUT EXT N、及びOUT VALID)に接続されたデバイスは、これらのトークンを有効なデータとして認識しないからである。以前と同様に、当該

ステージにおける双方の妥当性検査ラッチLVIN、LVOUTは、1つの単一コンダクタNOTRESET O、及び下流ラッチLVOUTの1つの単一リセット入力Rにより、上流の妥当性検査ラッチをその次のクロックサイクルにおいてローに強制させるように後方に伝播するリセット信号を用いてリセットされることが可能である。

【0212】図13、図14に示す例において、データトークンに含まれるデータの二重化は、回路がACCEPT及びVALID信号を操作可能であり、その結果として、当該入力に到達するよりもより多くのデータがパイプラインステージから離れることの一例としてのみ役立つに過ぎないことに注意されたい。同様に、流れからデータを除去するために回路がVALID信号を操作可能であることを示す方法を単に説明するだけのために、図13、図14に示す例においては、全ての非データトークンが除去される。ただし、最も典型的なアプリケーションにおいて、パイプラインステージは、当該パイプラインステージが認識しないあらゆるトークンを修正する事なく簡単に供給する。その結果、パイプラインの更に下流の他のステージが、必要に応じて、前記のトークンに作用することが可能である。図15及び図16は、図13及び図14に示すデータ二重化回路のためのタイミングダイアグラムの一例を示す。このタイミングダイアグラムは、2相クロック信号、様々な内部および外部制御信号、及びステージの入力側と出力側との間でデータをクロックし、そして、データを二重化する方法との間の関係を示す。さてここで、図17を更に詳細に参照することとし、この図には、現在の本発明の1つの態様にに基づく再構成可能なプロセスステージを示す。

【0213】入力ラッチ34は第1バス31を介して入力を受け取る。入力ラッチ34からの第1出力は、ライン32を介して、トークンデコードサブシステム33に供給される。入力ラッチ34からの第2出力は、ライン35を介して、第1入力としてプロセッシングユニット36に供給される。トークンデコード33からの第1の出力は、ライン37を介して、第2の入力として、プロセッシングユニット36に供給される。トークンデコード33からの第2出力は、ライン40を介して、アクション識別ユニット39に供給される。更に、アクション識別ユニット39は、ライン46を介してレジスタ43及び44から入力を受け取る。レジスタ43、及び44は、全体として、マシンの状態を保持する。この状態は、以前に受け取られたトークンのヒストリによって決定される。アクション識別ユニット39からの出力は、ライン38を介して、第3の入力としてプロセッシングユニット36に供給される。プロセッシングユニット36からの出力は、出力ラッチ41に供給される。出力ラッチ41からの出力は、第2バス42を介して供給される。

【0214】ここで、図18を参照することとし、ス

ートコード検出器 (SCD) 51 は、2 線インターフェース 52 を介して入力を受け取る。この入力は、データトークンの形式であるか、或いは、データの流れ内のデータビットであるかのいずれかである。スタートコード検出器 51 からの第 1 出力は、ライン 53 を介して、第 1 論理先入れ先出し方式バッファ (FIFO) 54 に供給される。第 1 FIFO 54 からの出力は、ライン 55 を介して、第 1 の入力として、ハフマンデコーダ 56 に論理的に供給される。スタートコード検出器 51 からの第 2 出力は、ライン 57 を介して、第 1 入力として、D 10 RAM インターフェース 58 に供給される。さらに、D RAM インターフェース 58 は、ライン 60 を介して、バッファマネージャ 59 から入力を受け取る。信号は、ライン 61 を介して、DRAM インターフェース 58 により、外部 DRAM (図示せず) に対して送信および受信される。DRAM インターフェース 58 からの第 1 出力は、ライン 62 を介して、第 1 物理入力として、ハフマンデコーダ 56 に供給される。ハフマンデコーダ 56 からの出力は、データ (ITOD) 64 までインデックスに入力としてライン 63 を介して回される。ハフマンデコーダ 56 及び ITOD 64 は、1 つの論理ユニットとして一緒に作動する。ITOD 64 からの出力は、ライン 65 を介して算術論理ユニット (ALU) 66 に供給される。ALU 66 からの第 1 出力は、ライン 70 を介して、読出し専用メモリ (ROM) ステートマシン 68 に供給される。ROM ステートマシン 68 からの出力は、ライン 69 を介して、第 2 物理入力として、ハフマンデコーダ 56 に供給される。ALU 66 からの第 2 の出力は、ライン 70 を介してトークンフォーマット部 (TF) 71 に供給される。

【0215】本発明にかかる TF からの第 1 出力 71 は、ライン 72 を介して第 2 の FIFO 73 に供給される。第 2 の FIFO 73 からの出力は、第 1 の入力として、ライン 74 を介して、逆モデラ 75 に供給される。T/F 71 からの第 2 の出力は、ライン 76 を介して、第 3 の入力として、DRAM インターフェース 58 に供給される。DRAM インターフェース 58 からの第 3 出力は、ライン 77 を介して、第 2 の入力として、逆モデラ 75 に供給される。逆モデラ 75 からの出力は、逆量子化器 79 への入力として、ライン 78 を介して供給される。逆量子化器 79 からの出力は、ライン 80 を介して、逆入力として、逆ジグザグ 81 (IZZ) に供給される。IZZ 81 からの出力は、ライン 82 を介して、離散逆コサイン変換 83 (IDCT) への入力として供給される。IDCT 83 からの出力は、ライン 84 を介して、時間デコーダ (図 19) に供給される。

【0216】ここで、更に詳細に図 19 を参照することとし、本発明に基づく時間デコーダがこの図に示される。フォーク 91 は、ライン 92 を介して、入力として、IDCT 83 (図 18) からの出力を受け取る。フ

ーク 91 からの第 1 出力として、例えば、モーションベクトル等のような制御トークンは、ライン 93 を介して、アドレス発生器 94 に供給される。更に、データトークンは、カウントする目的で、アドレス発生器 94 に供給される。このデータは、フォーク 91 からの第 2 の出力として、ライン 95 を介して FIFO 96 に供給される。次に、FIFO 96 からの出力は、ライン 97 を介して、第 1 の入力として加算器 98 に供給される。アドレス発生器 94 からの出力は、第 1 入力として、ライン 99 を介して、DRAM インターフェース 100 に供給される。信号は、ライン 91 を介して、DRAM インターフェース 100 により、外部 DRAM (図示されず) に対して、送信および受信される。DRAM インターフェース 100 からの第 1 出力は、ライン 102 を介して、予測フィルタ 103 に供給される。予測フィルタ 103 からの出力は、ライン 104 を介して、第 2 入力として、加算器 98 に供給される。加算器 98 からの第 1 出力は、ライン 105 を介して出力セクタ 106 に供給される。加算器 98 からの第 2 の出力は、ライン 107 を介して、第 2 の入力として DRAM インターフェース 100 に供給される。DRAM インターフェース 100 からの第 2 出力は、第 2 入力として、ライン 108 を介して、出力セクタ 104 に供給される。出力セクタ 104 からの出力は、ライン 109 を介して、動画フォーマット部 (図 20) に供給される。

【0217】ここにおいて、図 20 を参照することとし、フォーク 111 は、ライン 112 を介して、出力セクタ 106 (図 19) からの入力を受け取る。フォーク 111 からの第 1 の出力として、ライン 113 を介してアドレス発生器 114 は制御トークンを供給される。アドレス発生器 114 からの出力は、第 1 入力として、ライン 115 を介して DRAM インターフェース 116 に供給される。フォーク 111 からの第 2 出力としてのデータは、ライン 117 を介して、第 2 入力として、DRAM インターフェース 116 に供給される。信号は、ライン 118 を介して、DRAM インターフェース 116 により、外部 DRAM (図示されず) に対して、送信および受信される。DRAM インターフェース 116 からの出力は、ライン 119 を介して、表示パイプ 120 に供給される。

【0218】各ラインは、必要に応じて、複数のラインを有しても差し支えないことは、前述の説明から明白である。

【0219】ここにおいて、図 21 を参照することとし、MPEG 規格においては、1 つのピクチャ 131 は 1 つ又は複数のスライス 132 として符号化される。各スライス 132 は、複数のブロック 133 を有し、そして、各列において列毎に左から右に符号化される。図に示すように、各スライス 132 のスパンはブロック 133 3 の中の正確に 1 行 132 であるか、ブロック 133 の

中の1行より少ないB、Cか、またはブロック133の複数行Cであっても差し支えない。

【0220】図22を参照することとし、JPEG及びH. 261規格においては、コモン中間フォーマット

(CIF) が用いられ、ピクチャ141は、各列が2つのグループのブロック (GOB) 142を含む6列として符号化される。GOB 142は、3列または6列いずれかの限定されない数のブロック143によって構成される。各GOB 142は、矢印144によって示されるZ字形方向にジグザグに符号化される。その結果、GOB 142は、各列において、列毎に左から右に向かって処理される。

【0221】ここにおいて、図23を参照することとし、MPEG及びCIFの両方に関し、エンコーダの出力は、データストリーム151の形をとることが分かる。デコーダは、このデータストリーム151を受け取る。次に、デコーダは、イメージを符号化するために使われたフォーマットに応じてイメージを復元することができる。デコーダが、各規格に関するスタート及びエンドポイントを認識可能にするためには、データストリーム151は、33ブロック152の長さ分割される。

【0222】図24に示すベン図は、本発明のハフマンデコーダ56 (図18) からテーブル選択可能な値の領域を表す。これらの値は、MPEGデコーダ及びH. 261デコーダに対して重複可能な値であり、1つの単一テーブル選択によって、特定のMPEG及び特定のH. 261フォーマットの両方が復号化されることを示す。同様に、これらの値はMPEGデコーダ及びJPEGデコーダに対して重複可能な値であり、1つの単一テーブル選択によって、特定のMPEG及び特定のJPEGフォーマットの両方が復号化されることを示す。更に、H. 261値とJPEG値は重複せず、単独で双方のフォーマットを復号化するような単一テーブル選択が存在しないことを示す。

【0223】さて、図25を更に詳細に参照することとし、この図は、本発明の実施に基づいた可変長ピクチャデータの概要を示す。処理されるべき第1のピクチャ161は、第1ピクチャ・スタートトークン162、長さの限定されない第1ピクチャ情報163、及び第1ピクチャ・エンドトークン164を有する。処理されるべき第2のピクチャ165は、第2ピクチャ・スタートトークン166、長さの限定されない第2ピクチャ情報167、及び第2ピクチャ・エンドトークン168を有する。ピクチャ・スタートトークン162及び166は、ピクチャ161及び165のプロセッサに対するスタートを示す。同様に、ピクチャ・エンドトークン164及び168は、ピクチャ161及び165のプロセッサに対するエンドを意味する。これにより、プロセッサは、可変長さのピクチャ画像163及び167を処理することが可能となる。

【0224】図26において、スプリット171は、ライン172を介して入力を受け取る。スプリット171からの第1出力はライン173を介して、アドレス発生器174に供給される。アドレス発生器174によって生成されたアドレスは、ライン175を介して、DRAMインターフェース176に供給される。信号は、ライン177を介して、DRAMインターフェース176により、外部DRAM (図示されず) に対して、送信および受信される。DRAMインターフェース176からの第1出力は、ライン178を介して、予測フィルタ179に供給される。予測フィルタ179からの出力は、第1入力として、ライン180を介して、加算器181に供給される。スプリット171からの第2出力は、ライン182を介して、先入れ先出し方式バッファ (FIFO) 183への入力として供給される。FIFO 183からの出力は、ライン184を介して、加算器181への第2入力として供給される。加算器181からの出力は、ライン185を介して、ライト信号発生器186に供給される。ライト信号発生器186からの第1出力は、ライン187を介して、DRAMインターフェース176に供給される。ライト信号発生器186からの第2出力は、ライン188を介して、リード信号発生器189への第1入力として供給される。DRAMインターフェース176からの第2出力は、ライン190を介して、リード信号発生器189への第2入力として供給される。リード信号発生器189からの信号はライン191を介して動画フォーマッティング (図26には図示せず) へ供給される。

【0225】図27は、予測フィルタプロセスを示す。フォワード予測フィルタ201は、第1入力として、202を介して、加算器203に供給される。バックワード予測フィルタ204は、第2入力として、ライン205を介して、加算器203に供給される。加算器203からの出力は、ライン206を介して供給される。図28に示すように、スライス211は、1又は複数のマクロブロック212を有する。結果的に、各マクロブロック212は、4つのルミナンス輝度ブロック213、及び2つのクロミナンスブロック214を有し、そして、ピクセルのものの16×16ブロックに関する情報を有する。4つのルミナンスブロック213及び2つのクロミナンスブロック214の各々のサイズは8×8ピクセルである。4つのルミナンスブロック213は、1ピクセル画素をピクセルのものの16×16ブロックのルミナンス (Y) 情報の1ピクセル毎のマッピングを有する。1つのクロミナンスブロック214は、青色信号 (Cu/b) のクロミナンスレベルの表示を含み、そして、いま1つのクロミナンスブロック214は、赤色信号 (Cv/r) のクロミナンスレベルの表示を含む。各クロミナンスレベルは、各8×8クロミナンスブロック214が、ピクセルのものの16×16ブロック全体に

関するその色信号のクロミナンスレベルを含むように、サブサンプルされる。

【0226】図29を参照することにより、スタートコード検出器の構造及び機能が明白になるはずである。バリュレジスタ221は、ライン222を介してイメージデータを受け取る。ライン222の幅は8ビットであり、一度に8ビットの並列伝送を可能にする。バリュレジスタ222からの出力は、ライン223を介して、デコードレジスタ224に直列に供給される。デコードレジスタ224からの第1の出力は、ライン226を介して、検出器225に供給される。ライン226の幅は24ビットであり、一度に24ビットの並列伝送を可能にする。検出器225は、1つの「1」バリュによって後続される23の「ゼロ」バリュの規格から独立したスタートコードに対応するイメージの存在または欠如を検出する。8ビットデータバリュイメージは、有効なスタートコードイメージに後続する。スタートコードイメージの存在が検出されると、検出器225は、ライン227を介してスタートイメージをバリュデコーダ228に伝送する。

【0227】デコードレジスタ224からの第2出力は、ライン229を介して、バリュデコードシフトレジスタ230に直列に供給される。バリュデコードシフトレジスタ230は、ビット長15の1つのデータバリュイメージを保持することができる。部分領域231によって示されるように、スタートコードイメージに後続する8ビットデータバリュは、バリュデコードシフトレジスタ230の右にシフトされる。以下において検討するように、このプロセスは、スタートコードイメージの重複を排除する。バリュデコードシフトレジスタ230からの第1出力は、ライン232を介して、バリュデコーダ228に供給される。ライン232の幅は15ビットであり、一度に15ビットの並列伝送を可能にする。バリュデコーダ228は、第1のルックアップテーブル（図示されず）を用いてバリュイメージを復号化する。バリュデコードシフトレジスタ230からの第2出力は、ライン235を介してインデックス/トークンコンバータ234にフラグを供給するバリュデコーダ228に供給される。更に、バリュデコーダ228は、ライン236を介して、インデックス/トークンコンバータ234に情報を供給する。情報は、第1ルックアップテーブルから入手されるデータバリュイメージ、或いは、スタートコードインデックスイメージのいずれかである。フラグは、情報が供給されるフォームを示す。ライン236の幅は15ビットであり、一度に15ビットの並列伝送を可能にする。本発明においてこの場合における幅として15ビットが選定されたが、他の長さのビットを使用しても差し支えないことが理解されるはずである。インデックス/トークンコンバータ234は、ユーザ用マニュアルの表12-3記載の

場合と同様の第2ルックアップテーブル（図示されず）を用いて情報をトークンイメージに変換する。インデックス/トークンコンバータ234によって生成されたトークンイメージは、ライン237を介して出力される。ライン237の幅は15ビットであり、一度に15ビットの並列伝送が可能である。

【0228】図30において、個々のビット242から成るデータストリーム241は、スタートコード検出器（図30に図示されず）に入力される。第1スタートコードイメージ243は、スタートコード検出器によって検出される。次に、スタートコード検出器は、第1データバリュイメージ244を受け取る。第1データバリュイメージ244を処理する以前に、スタートコード検出器が、長さ246において第1データバリュイメージ244と重複する第2スタートコードイメージ245を検出することもあり得る。これが起きると、スタートコード検出器は第1データバリュイメージ244を処理せず、そして、その代わりに、第2データバリュイメージ247を受け取り、そして、処理する。

【0229】図31において、フラグ発生器251は、ライン252を介して第1入力としてのデータを受け取る。ライン252の幅は15ビットであり、一度に15ビットの並列伝送を可能にする。同様に、フラグ発生器251は、ライン253を介して、第2の入力としてフラグを受け取り、そして、第1の2線インターフェース254を介して入力有効イメージを受け取る。フラグ発生器251からの第1出力は、ライン255を介して、入力有効レジスタ（図示されず）に供給される。フラグ発生器251からの第2出力は、ライン256を介して、デコードインデックス257に供給される。デコードインデックス257は次の4つの出力を生成する。即ち、ピクチャスタートイメージはライン258を介して出力され、ピクチャナンバイメージはライン259を介して出力され、インサートイメージはライン260を介して出力され、そして、リブレースイメージはライン261を介して出力される。フラグ発生器251からのデータはライン262aを介して供給される。ヘッダ発生器263は、リブレースイメージを生成するためにルックアップテーブルを使用し、このリブレースイメージは262bを介して供給される。エクストラワード発生器264は、インサートイメージを生成するためにMPUを使用し、インサートイメージはライン262cを介して供給される。ライン262aは、出力ラッチ265への第1入力である262ラインを形成するために結合する。出力ラッチ265は、ライン266を介してデータを出力する。ライン266の幅は15ビットであり、一度に15ビットの並列伝送を可能にする。

【0230】入力有効レジスタ（図示されず）は、ライン268を介して、第1ORゲート267への第1入力としてイメージを供給する。インサートイメージは、ラ

57

イン269を介して、第1ORゲート267への第2入力として供給される。第1ORゲート267からの出力は、ライン271を介して、第1ANDゲート270への第1入力として供給される。リムーブイメージの論理否定は、ライン272を介して、第1ANDゲート270への第2入力として供給され、そして、ライン273を介して、出力ラッチ265への第2入力として、供給される。出力ラッチ265は、第2の2線インターフェース274を介して出力有効イメージを供給する。出力アクセプトイメージは、第2の2線インターフェース274を介して、出力アクセプトラッチ275により受け取られる。出力アクセプトラッチ275からの出力は、ライン276を介して、出力アクセプトレジスタ（図示されず）に供給される。

【0231】出力アクセプトレジスタ（図示されず）

表600

フォーマット	受信画像	トークン
1 H. 261	シーケンススタート	シーケンススタート
MPEG	ピクチャ・スタート	グループスタート
JPEG	無し	ピクチャ・スタート
		ピクチャデータ
2 H. 261	無し	ピクチャエンド
MPEG	無し	パディング
JPEG	無し	フラッシュ
		ストップ・アフタ・ピクチャ

特定マシン独立制御トークンにおける標準信号の存在と欠如との間の関係を示す表600に示すように、スタートコード検出器51によるイメージの検波は一連のマシン独立制御トークンを生成する。「受信イメージ」カラムにリストされた各イメージは、「生成トークン」カラムのグループにリストされた全てのマシン独立制御トークンの生成をスタートする。表600の1行目に示すように、H. 261処理期間中に「シーケンススタート」

表601

表示順序	I 1	B 2	B 3	P 4	B 5	B 6	P 7	B 8	B 9	I 10
送信順序	I 1	P 4	B 2	B 3	P 7	B 5	B 6	I 10	B 8	B 9

伝送されたピクチャと表示されたピクチャとの間のタイミング関係を示す表601の1行に示すように、ピクチャフレームは、番号順にディスプレイされる。ただし、メモリに記憶しなければならないフレームの数を減少するために、フレームは、異なる順序で送信される。イントラフレーム（Iフレーム）から分析を始めることは有益である。I1フレームは、ディスプレイされるべき順序に送信される。次に、その次の予測されるフレーム（Pフレーム）P4が送信される。次に、I1フレームとP4フレームとの間にディスプレイされるべき双方向的に補間されるあらゆるフレーム（Bフレーム）が送信される。これらのフレームをフレームB2及びB3によって表す。これにより、送信済みのBフレームに、前のフレーム（フォワード予測）または将来のフレーム（バックワード予測）を参照させることが可能にする。

は、ライン278を介して、第1入力として、第2ORゲート277にイメージを供給する。入力有効レジスタからの出力の論理否定は、ライン279を介して、第2ORゲート277への第2入力として供給される。リムーブイメージは、ライン280を介して、第3入力として、第2ORゲート277に供給される。第2ORゲート277からの出力は、ライン282を介して、第2ANDゲート281への第1入力として供給される。インサートイメージの論理否定は、ライン283を介して、第2ANDゲート281への第2入力として供給される。第2ANDゲート281からの出力は、ライン284を介して、入力アクセプトラッチ285へ供給される。入力アクセプトラッチ285からの出力は、第1の2線インターフェース254を介して供給される。

イメージが受け取られるか、或は、PEC処理期間中に「ピクチャ・スタート」イメージが受け取られる場合にはいつでも、4つの制御トークンの全グループが生成され、各々がその対応する1つまたは複数のデータバリュエによって後続される。更に、表600の2行目に示すように、4つの制御トークンの第2グループは、スタートコード検出器51によって受け取られるイメージにかかわらず適切な時間に生成される。

I1フレームとP4フレームとの間にディスプレイされるべき全てのBフレームを送信した後で、次のPフレームであるP7が送信される。次に、P4とP7フレームとの間にディスプレイされるべきB5及びB6に対応する全てのBフレームが送信される。次に、その次のIフレームであるI10が送信される。最終的に、P7とI10フレームとの間にディスプレイされるべきフレームB8及びB9に対応する全てのBフレームが送信される。この順序でフレームを送信するためには、あらゆる一時にただ2つのフレームをメモリに保持することが必要であり、そして、中間のBフレームをディスプレイするために次のPフレーム又はIフレームの送信をデコードに待機させることを必要としない。

【0232】本発明の構造及びオペレーション、並びに、特徴、目的、及び利点に関するこれ以上の情報は、当該技術分野における通常の熟練者であれば、本発明の例証的な実施例について追加的に継続される詳細な記述から容易に明白になるはずである。この詳細な内容については、説明を明瞭かつ利便にするために、次に示すセクションにまとめて記述することとする。

【0233】1. 多重規格コンフィギュレーション

2. J P E G 静止画のデ符号化

3. 動画の伸長

4. R A M メモリマップ

5. ビットストリームの特性

6. 再構成可能なプロセッシングステージ

7. 多重規格符号化

8. 多重規格プロセッシング回路 — 第2オペレーションモード

9. スタートコード検出器

10. トークン

11. D R A M インターフェース

12. 予測フィルタ

13. アクセシングレジスタ

14. マイクロプロセッサインターフェース (M P I)

15. M P I リードタイミング

16. M P I ライトタイミング

17. キーホールアドレスロケーション

18. ピクチャエンド

19. フラッシングオペレーション

20. フラッシュ機能

21. ピクチャ後ストップ

22. 多重規格サーチモード

23. 逆モデラ

24. 逆量子化器

25. ハフマンデコーダ及びパーザ (parser) (構文解析系)

26. 種々の別個のコサイントランスフォーマ

27. バッファマネージャ

1. 多重規格コンフィギュレーション

前述の米国特許第 5, 212, 742 号明細書に示したように、様々な圧縮規格、即ち、J P E G、M P E G、及び H. 261 が周知であるので、これらの規格の詳細な仕様について、ここでは繰り返さない。

【0234】以前に言及したように、本発明は、種々異なってコード化された様々なピクチャデータビットストリームを伸長することが可能である。コード化に関する異なる各規格において、単独で作動する空間デコーダの出力、或いは、組合わされて作動する空間デコーダ及び時間デコーダの直列出力に供給されたデータを扱い (続いて更に詳細に説明されるように)、そして、コンピュータにおけるディスプレイ又は動画ディスプレイシステムを含む他のディスプレイシステムを含めて使用するた

めにこの出力をリフォーマットするには、或る程度規制された形の出力フォーマット部が必要である。このフォーマット部を実現する方法は、コード化規格、及び/又は、選定されたディスプレイのタイプの間で大幅に変化する。

【0235】本発明に基づく第1の実施例においては、図17~図19に関して以前に述べたように、フォーマット化されたデータのブロック、即ち、第1のデコーダ (空間デコーダ)、又は、第1のデコーダ (空間デコーダ) と第2のデコーダ (時間デコーダ) との組み合わせのどちらかからの出力を記憶し、そして、複写された情報を、ラスト順に、メモリーに対して書き込み、及び/又は、書き出すために、アドレス発生器が用いられる。以下に説明する動画フォーマット部は、広い範囲に亘る出力信号組合わせを提供する。

【0236】本発明にかかる好ましい多重規格動画デコーダ実施例において、空間デコーダ及び時間デコーダは、M P E G 符号化信号および H. 261 動画符号復号化システムの両方を実現することを必要とする。これら双方のデバイスの D R A M インターフェースは、小さいピクチャフォーマットを用い、そして、低いコード化データレートにおいて運用している場合には、必要とされる D R A M の量を少なくするために構成可能である。これらの D R A M の再構成については、D R A M インターフェースに関連して以下において更に説明することとする。一般に、各々の時間デコーダ及び空間デコーダ回路によって1つの単一4メガバイト D R A M が必要とされる。

【0237】本現在の空間デコーダは、1つの単一ピクチャ内で必要とされる全ての処理を遂行する。これにより、1つのピクチャ内の冗長度が減少される。

【0238】時間デコーダは、主体とされるピクチャの到着前に到着するピクチャに関係する主体となる当該ピクチャ、並びに、主体とされるピクチャの到着後に到着するピクチャとの間の冗長性を減少させる。時間デコーダの1つの態様は、これら全てのピクチャに関連したデータを読み取るために必要な複雑なアドレス機能を、出来るだけ少ない個数の回路を使用し、そして、高速および改良された精度を以て、扱うアドレスデコードネットワークを提供することである。

【0239】図18に関して以前に説明したように、データは、ハフマンデコーダ及びパーザに先行するスタートコード検出器、F I F O、第2の F I F O レジスタ、逆モデラ、逆量子化器、逆ジグザグ及び逆 D C T を介して到着する。2つの F I F O は、チップ上に所在する必要はない。一実施例において、データは、チップ上に所在する F I F O を通って流れない。データは、D R A M インターフェースに供給され、そして、F I F O - I N 記憶レジスタ及び F I F O - O U T レジスタは、両方の場合に、チップ外に所在する。オペレーションが

規格とは完全に独立しているこれらのレジスタについては、続いて更に詳細に説明する。

【0240】図18に示すサブシステム及びステージの大多数は、用いられる特定の規格から実際に独立しており、そして、DRAMインターフェース58、DRAMインターフェースのためにアドレスを生成するバッファマネージャ59、逆モデラー75、逆ジグザグ81、及び逆DCT83を含む。ハフマンデコード及びパーザ内の規格から独立したユニットは、ALU66及びトランクフォーマッティング部71を含む。

【0241】図19において、規格から独立したユニットには、DRAMインターフェース100、フォーク91、FIFOレジスタ96、加算器98、及び出力セレクタ106が含まれる。規格従属ユニットは、H. 261とMPEGにおいて異なるアドレス発生器94、及びH. 261及びMPEGの両方に適合するように再構成可能な予測フィルタ103である。JPEGデータは、一切変更されない機械全体を通して流れる。

【0242】図20は、動画フォーマッティング部チップのハイレベル構成図を示す。このチップの殆どの部分は規格から独立している。規格によって影響される僅かな項目は、MPEG又はJPEGと異なるH. 261の場合にはDRAMにデータが記入される方法であり、そして、H. 261においては、全ての単一ピクチャをコード化する必要はない。ピクチャを何時ディスプレイしようとするかに関するいくつかの情報を提供し、そして、動画フォーマッティング部におけるロジックのアドレス発生タイプによっても扱われる時間基準として参照されるいくつかのタイミング情報がある。

【0243】動画フォーマッティング部に具現された回路の残りの部分は、色空間変換の全て、アップサンプリングフィルタ、及びガンマ補正RAMの全てを含み、そして、用いられる特定の圧縮規格から完全に独立している。

【0244】本発明のスタートコード検出器は圧縮規格に依存し、この圧縮規格において、前記検出器は、当該規格の各々に対して、ビットストリームにおける種々の異なるスタートコードパターンを認識しなければならない。例えば、H. 261は16ビットスタートコードを持ち、MPEGは24ビットスタートコードを持ち、そして、JPEGは他のスタートコードとはかなり異なるマーカコードを使用する。いったん、スタートコード検出器がこれらの異なるスタートコードを認識してしまえば、そのオペレーションは、圧縮規格から本質的に独立する。例えば、種々の異なるカテゴリのマーカを認識する回路は別として、オペレーションの多くは、3つの異なる圧縮規格の間で非常に類似している。

【0245】次に示すユニットは、ハフマンデコード及びパーザ内に位置するステートマシン518(図18)である。この場合、実際の回路構成は、3つの圧縮

規格の各々に関してほとんど同じである。実際、オペレーションにおいて規格によって影響される唯一の要素は、マシンのリセットアドレスである。単にパーザがリセットされる場合には、パーザは、各規格に関して異なるアドレスへジャンプする。実際、認識される4つの規格がある。これらの規格は、H. 261、JPEG、MPEG、及び他の1つ規格であり、最後の規格の場合、パーザは、テストのために使われる1つのコードに入る。この事実は、回路がほとんど全ての態様において同じであり、各々の規格に関するマイクロコードによるプログラムが異なるだけである。従って、1つのプログラムがH. 261において作動しており、そして、別の異なる1つのプログラムがランしている場合、これらのプログラムの間には重複はない。これは、JPEGに関しても真であり、第3の完全に独立したプログラムである。

【0246】次に示すユニットは、データインデックスユニット64と共に機能するハフマンデコード56である。これら2つのユニットは、ハフマン復号化を遂行するために、協同作動する。この場合、ハフマン復号化のために使われるアルゴリズムは、圧縮規格に関係なく同じである。変化は、どちらの表が用いられるかということ、および、ハフマンデコードに到来するデータが反転されているかどうかということである。更に、ハフマンデコード自体、符号化規格のいくつかの態様を理解するステートマシンを含む。これらの異なるオペレーションは、パーザステートマシンから到来する命令に回答して選定される。パーザステートマシンは、3つの圧縮規格の各々に対して異なるプログラムによって作動し、そして、作動中の規格に適合する異なる時点において、正しい命令をハフマンデコードに供給する。

【0247】圧縮規格に依存するチップ上の最後のユニットは、逆量子化器79であり、この場合、逆量子化器が遂行する数学は異なる規格の各々に対して異なる。この際に、コーディング・スタンダードトークンは復号化され、そして、逆量子化器79は、どちらの規格の下で作動中であるかを記憶する。次に、当該イベントの後に起き、他のコーディング・スタンダードの前に起きかもしれないあらゆる後続データトークンは、逆量子化器内に記憶されているコーディング・スタンダードによって示された方法によって扱われる。詳細な記述において言及する表は、異なる規格における異なるパラメータを示し、これらの異なるパラメータ又は数学にどの回路が対応するかを示す。

【0248】H. 261に基づくアドレス生成は、図19及び図20に示す各々のサブシステムに対して同じではない。図18に示すハフマンデコードの前或いは後の2つのFIFOに対してアドレスを生成するアドレス生成方法は、コード化規格に応じて変化しない。H. 261の場合であっても、当該チップ上で起きるアドレス生成

は変わらない。本質的に、これらの規格の間の差は、MPEGとJPEGとの差であり、マクロブロックの組織は、ピクチャを横断して水平に伸びる線状の行内に配置される。図21に示すように、第1のマクロブロックAが1行をカバーする。マクロブロックBは1行未満をカバーする。マクロブロックCは複数行をカバーする。MPEGにおいては、スライスに分割され、そして、1つのスライスが1つの水平行Aである場合もあり、或いは、1つのスライスが水平行Bの一部である場合もあり、或いは、1つのスライスが1つの行からその次の行に伸びる場合、Cもあり得る。これらのスライスの各々は、マクロブロックの列で構成される。

【0249】H. 261の場合には、ピクチャはブロックのグループ(GOB)に分割されるので、組織はいくぶん異なる。ブロックのグループは、高さが3列のマクロブロックに相当し、幅が11マクロブロックに相当する。CIPピクチャの場合には、この種の12のブロックグループがある。ただし、1つの上に他の1つと言うように組織されていない。そうではなくて、高さ6ブロックのグループが前後2グループ配置される。即ち、垂直に6GOB及び水平2GOBが配置される。

【0250】他の全ての規格において、アドレッシングを遂行する場合、既に示したように、マクロブロックは順番にアドレスされる。更に詳細には、アドレスは行に沿って進行し、そして、行の終端において、その次の行がスタートされる。H. 261の場合には、ブロックの順序は、ブロックのグループ内において記述された順序と同じであるが、しかし、その次のブロックグループに動く際にそれは、ほとんどジグザグである。

【0251】本発明は、前記の影響を扱うための回路を提供する。それは、空間デコード及び動画フォーマティング部におけるアドレス生成を、H. 261用に変更する方法である。情報がDRAMに記入されるときはいつでも、これが達成される。前述のアドレス生成シーケンスに関する知識によって記入されるので、RAMにおける物理的に配置される場所は、同じサイズのMPEGピクチャの場合と厳密に同じである。従って、メモリーにおける情報の物理的配置は、MPEGシーケンスであった場合と同じであるので、DRAMから読取りを行うための全てのアドレス生成回路は、例えば予測を形成する場合、H. 261規格の場合であることを理解する必要はない。このように、全ての場合に、データの記入のみが影響される。

【0252】時間デコードにおいて、回路が、実際に発生している事柄と異なる何かであるように見えるH. 261に対して抽象化が行われる。即ち、ブロックの各グループは概念的に拡張され、その結果、11×3マクロブロックである長方形の代わりに、マクロブロックは、高さが1マクロブロックであり長さが33ブロック(図23)のブロックグループに拡張される。こうすること

により、ブロックのグループ通じてカウントするために時間デコードにおいて用いられるカウント技法と全く同じ技法が、MPFG用としても用いられる。

【0253】ブロックのH. 261グループとMPEGスライスとの間に合致が存在するように回路が設計される。スタートコード検出器の後でH. 261データが処理される場合、各ブロックグループは、スライススタートコードによって先行される。その次のブロックグループは、その次のスライススタートコードによって先行される。この構造を通してカウントするために時間デコード内においてカウントが行われる場合は、高さが1マクロブロックであって長さが33マクロブロックのグループであるかのように見える。同様に回路は10間隔おきにカウントするが、これで十分である。第11番目のマクロブロック又は第22番目のマクロブロックをカウントすると、幾つかのカウンタをリセットする。各マクロブロックをカウントアップし、そして、11に到達すると、ゼロにリセットする別のカウンタを備えた簡単な回路によってこれが達成される。マイクロコードは、それを尋問し、そして、その作用を実施する。本発明の時間デコードにおける全ての回路は、マクロブロックの物理的配置に関して、圧縮規格から本質的に独立している。

【0254】多重規格の適応性に関して多数の異なる表があり、そして、回路は、適切な時点において適切な規格を適応するために適切な表を選定する。各規格は複数の表を持ち、そして、回路は、任意の所定時点において表の集合の中から選定する。任意の1つの規格内において、回路は、一時に1つの表を選定し、また、別の一時に別の表を選定する。異なる規格内においては、回路は、異なる表の集合を選定する。図24に関する検討の際に既に指摘したように、これらの表の間には或る程度の交差がある。例えば、MPEGに用いられる表の1つは、JPEGにおいても同様に用いられる。これらの表は、完全に分離した集合ではない。図24は、H. 261集合、MPEG集合、及びJPEG集合を示す。H. 261集合とMPEG集合との間には非常に大きい重複があることに注意されたい。これらの規格が使用する表は全く共通する。MPEGとJPEGとの間には僅かな重複があり、そして、H. 261とJPEGとの間には全く重複がないので、これらの規格は完全に異なる表の集合を持つ。

【0255】既に指摘したように、大部分のシステムユニットは圧縮規格から独立している。或る1つのユニットが規格独立的である場合には、この種のユニットは、どのコーディング・スタンダードが処理されつつあるかを記憶している必要がない。規格従属的な全てのユニットは、コーディング・スタンダードトークンがこれらのユニットを流れる際に、圧縮規格を記憶している。第1の符号化規格においてコード化/復号化された情報はマシン全体に配分され、そして、マシンが規格を変更しつ

つある場合には、マイクロプロセッサ制御される前のマシンは、一般に、H. 261 圧縮規格に従って機能することを選定する。この種の前のマシンにおけるMPUは、圧縮規格が変更中のマシン内における複数の異なる場所において信号を生成する。MPUは、異なる時点において変更を行い、そしてパイプライン全体を通じてフラッシュする。

【0256】本発明に従い、パイプラインにおける第1のユニットとして配置されたスタートコード検出器においてコーディング・スタンダードトークンの変化を起こさせることにより、圧縮規格のこの変化を容易に取り扱うことができる。トークンは、あるコード化規格が開始されつつあることを表明し、そして、当該制御情報は、マシンを下流に向かって流れ、そして、適切な時点において他の全てのレジスタを構成する。MPUは、各レジスタをプログラムする必要がない。

【0257】予測トークンは、ビットストリーム内のビットを用いて予測を形成する方法を告げる。どの圧縮規格が運用中であるかに応じて、回路は、当該規格内に発見される情報をビットストリームから予測モードトークンへ変換する。この処理は、ハフマンデコード、及びパーザーステートマシンによって行われ、この場合に、特定の条件に基づくビットを操作することは容易である。スタートコード検出器は、この予測モードトークンを生成する。次に、トークンは、マシンを下流に向かって、予測形成に責任のあるデバイスである時間デコードの回路まで流れる。ある規格内のビットは3つの異なる規格内において変化しないので、空間デコードの回路は、トークンがその中で運用されている規格がどの規格であるかを知ることなしに、当該トークンを解釈する。空間デコードは、当該トークンに回答して告げられたことだけを行う。これらのトークンを所有し、そして、それらを適切に使うことによって、マシンにおける他のユニットの設計が簡素化される。プログラムは幾分複雑になることもあり得るが、多重規格に対して設計困難なハードワイヤロジックをこの場合に使用することができる利益が得られる。

【0258】2. JPEG 静止画像の復号化

以前に指摘したように、本発明は信号伸長に関し、更に詳細には、使用圧縮規格には無関係なコード化動画信号の伸長に関する。

【0259】本発明の1つの態様は、パイプライン処理システムにおいて、第1の符号化された信号(MPEG、またはH. 261 符号化動画信号)を復号化するために第2のデコード回路(時間デコード)と組み合わせる第1の符号化された信号(JPEG 符号化動画信号)を復号化するための第1のデコード回路(空間デコード)を提供することである。JPEG 符号復号化のためには時間デコードを必要としない。

【0260】この点に関して、本発明は、1つの単一パ

イプラインデコード及び伸長システムを使用することにより異なる方法で符号化された複数の信号の伸長を容易にする。符号復号化および伸長パイプラインプロセッサは、独特の、そして、単一パイプラインデコード及び処理システムと完全に互換性のある技術を用いて多重規格符号化動画信号の扱いを可能にする特殊なコンフィギュレーションに組織されている。空間デコードは時間デコードと組合わされ、そして、動画フォーマット部は、動画ディスプレイをドライブするために用いられる。

【0261】本発明の他の態様は、静止画のみと使用するために、空間デコードと動画フォーマット部の組み合わせを使用することである。圧縮規格から独立した空間デコードは、1つの単一ピクチャの境界内において全てのデータ処理を遂行する。この種のデコードは、パイプラインを通過し、そして、関連している等速呼出記憶装置、メモリーに対して情報の記憶および検索を扱うための規格から独立したアドレス生成回路内に配分される内部ピクチャデータの空間伸長を扱う。静止画像データは、空間デコードの出力において復号化され、そして、この出力は、ディスプレイ端末に出力を供給する多重規格構成可能動画フォーマット部の入力として用いられる。同様の画像の第1のシーケンスにおいて、画像が空間デコードの出力に到達するまでは、空間デコードの出力における伸長された各画像は同じビット長さである。画像の第2シーケンスは、完全に異なる画像サイズであっても差し支えなく、従って、第1の長さと比較した場合に、長さが異なっても差し支えない。再度説明すれば、この種画像が空間デコードの出力に到達するまでは、類似した画像のこの種の第2シーケンスは全てビット長さが同じである。

【0262】本発明の他の態様は、入来する規格従属ビットストリームを、規格から独立した再構成可能なパイプラインプロセッサとして作動するように選定および組織化され、順序立てて配置された再構成可能な複数の処理ステージと組合わされた一連の制御トークン及びデータトークンに内部的に組織化することである。

【0263】JPEG 復号化に関して、チップ外DRAM無し1つの単一空間デコードは、基底線JPEG イメージを急速に復号化することが出来る。空間デコードは、基底線JPEG 符号化規格の全ての機能をサポートする。ただし、復号化可能なイメージサイズは、装備された出力バッファのサイズによって制限されることもある。空間デコード回路は、情報のメモリーへの記憶を扱うための機械従属、規格から独立したアドレス生成回路を有する等速呼出記憶装置回路を含む。

【0264】既に指摘したように、JPEG 符号化された動画を復号化するためには時間デコードを必要としない。従って、データトークンによって運ばれる信号は、JPEG オペレーション用として時間デコードが構成さ

れている場合には、それ以上処理することなしに、時間デコーダを通して直接供給される。

【0265】本発明の他の態様は、ハフマンデコーダ／動画デマルチプレクサ回路（HD & VDM）と組み合わせられて作動するための、例えばバッファメモリ回路のような、1対のメモリ回路を空間デコーダ内に提供することである。第1の緩衝記憶装置は、HD & VDMの前に配置され、そして、第2の緩衝記憶装置は、HD & VDAの後に配置される。KD & VDMは、2進1及び0出構成されるビットストリームを復号化する。これらの数値は、規格符号化されたビットストリーム内に所在し、そして、当該ビットストリームを、下流において使用される数に変換する。2バッファシステムの利点は、多重伸長システムを実現するために使用できることである。これら2つのバッファについては、ハフマンデコーダの確認済み実施例と組み合わせ、この後で詳細に説明される。

【0266】本多重規格伸長回路の更に別の態様は、ハフマンデコーダと組み合わせ、作動する第1のフォワードバッファの上流に配置されたスタートコード検出器回路と組合わされることである。この組み合わせの1つの利点は、入力ビットストリーム、特にパディングを扱う場合に、ビットストリームに加えられなければならない融通性が増大することである。これらの確認済みコンポーネント、スタートコード検出器、メモリバッファ、及びハフマンデコーダを配置すると、入力ビットストリームにおける特定のシーケンスの扱いを強化する。

更に、チップ外DRAMは、JPEG符号化された動画画像をリアルタイムに復号化するために用いられる。DRAMと共に使われるバッファのサイズ及び速度は、動画符号化されたデータレートに依存する。

【0267】符号化規格は、規格から独立した回路を用いる空間デコーダと関連するDRAMに記憶するために必要な規格従属タイプの全ての情報を識別する。

【0268】3. 動画伸長

本発明において、復号化過程を経て動画が伸長されつつある場合には、更に、時間デコーダが必要である。時間デコーダは、空間デコーダにおいて復号化されたデータを現在復号化されつつあるピクチャ像の後でディスプレイすることを意図されている既に復号化済みの画像と組み合わせる。時間デコーダは、画像符号化データストリームにおいて、この時間的に置き換えられた情報を識別するための情報を受け取る。時間デコーダは、時間的および空間的に配置された情報をアドレスし、情報を検索し、現在復号化されつつあり、そして、完了し、そして、ディスプレイスクリーンをドライブするために動画フォーマッティング部への伝送に適した結果的な画像を終了させる画像を用いて1つの画像内に配置された情報を復号化する方法によって情報を組み合わせる。その代りに、結果的な画像は、後続する画像の時間的復

号化のためにその後で使用するために記憶しておくことができる。

【0269】一般に、時間デコーダは、現在復号化されつつある画像に対して早期及び／又は後期のいずれかに於いて画像間の処理を実施する。時間デコーダは、当該画像の符号化された表示内に符号化されない情報を再導入する。理由は、当該情報が冗長であり、そして、デコーダにおいて既に利用可能であることに因る。更に詳細には、あらゆる所定の画像が、前後両方において当該情報を時間的に囲むような類似の情報を含むことが可能である。運動補償が適用されるならば、この類似性を更に大きくすることができる。時間デコーダ及び伸長回路は、同様に、関連した画像の間の冗長度を減少させる。

【0270】本発明の他の態様において、時間デコーダは、空間デコーダからの規格従属出力情報を扱うために用いられる。1つの単一画像に関するこの規格従属情報は、空間デコーダによって処理された伸長された出力情報が、第1の画像の時間的な位置に関して時間的に配置された空間的に復号化された画像情報の空間的に復号化された情報パケットの1つの画像を組み合わせるために、更に他のマシンに従属し規格から独立したアドレス生成回路を有する他の等速呼出記憶装置によって他のDRAMレジスタに記憶されるような方法において、DRAMの幾つかの領域の間に配分される。

【0271】MPEG符号化された信号を復号化することの出来る多重規格回路においては、更に大きい論理DRAMバッファは、MPEGに適合可能な更に大きい画像フォーマットをサポートすることが要求される場合もあり得る。

【0272】画像情報は、8画素×8画素ブロックにおける直列パイプラインを通して移動中である。本発明の1つの形において、アドレス復号化回路は、ブロック境界に沿ってこれらの画素ブロックを扱う（記憶および検索する）。更に、アドレス復号化回路は、境界を横断して行われる8×8画素ブロックの記憶および検索を扱う。この汎用性については、以下において更に完全に説明することとする。

【0273】第2の時間デコーダは、同様に、信号処理遅延なしに扱うために、第1のデコーダ回路（空間デコーダ）の出力を直接動画フォーマッティング部に供給しても差し支えない。

【0274】更に、時間デコーダは、ディスプレイ回路による遅延に対して、画像データのブロックを順序付けし直す。後で説明するように、アドレス復号化回路はこの順序付けし直しを実施可能にする。

【0275】既に述べたように、時間デコーダの1つの重要な特徴は、画像の選択から得られた処理中の画像よりも早期に、或いは、遅れて到着した画像情報を合計することである。

【0276】この文脈においてピクチャについて記述す

る場合、ピクチャは次に示す内容のいずれかを意味する。

【0277】1. ピクチャの符号化されたデータ表示である。

【0278】2. 結果である、即ち、デコーダによって行われたプロセス段階を加算した結果として得られる最終的な復号化済みのピクチャである。

【0279】3. DRAMから読み出された既に復号化済みのピクチャである。

【0280】4. 空間的復号化の結果である。即ち、ピクチャ・スタートトークンとその次のピクチャ・エンドトークンとの間のデータの範囲である。

【0281】ピクチャデータ情報が時間デコーダによって処理された後で、情報は、ピクチャ記憶場所にディスプレイされるか、或いは、書き戻される。次に、この情報は、別の異なる符号化済みデータピクチャを処理する際に用いられる更なる基準として保持される。視覚的ディスプレイのための MPEG 符号化されたピクチャの再配列には、可所要のスクランブルされたピクチャは、時間デコーダの再配列機能を変えることによって達成される能性が包含される。

【0282】4. RAMメモリーマップ

空間デコーダ、時間デコーダ、及び動画フォーマット部は、全て、外部DRAMを使用する。これら全ての3つのデバイスには同じDRAMが用いられることが好ましい。これら3つ全てのデバイスがDRAMを用い、更に、3つ全てのデバイスがアドレス発生器と共にDRAMインターフェースを使用する場合であっても、各々がDRAMにおいて実現するものは同じでない。即ち、各チップ、例えば、空間デコーダ及び時間デコーダは、同じ物理的な外部DRAMを使用する場合であっても、異なるDRAMインターフェース及びアドレス生成回路を有する。

【0283】要するに、空間デコーダは、共通DRAM内に2つのFIFOを実現する。再び図18を参照することとし、一方のFIFO54は、ハフマンデコーダ56及びパーザーの前に配置され、そして、他方は、ハフマンデコーダ及びパーザーの後に配置される。FIFOは、比較的直截な方法において実現される。DRAMの特殊な部分は、各FIFOに対して、その中にFIFOを実現するための物理的メモリーとして控除しておかれる。

【0284】空間デコーダDRAMインターフェース58と関連しているアドレス発生器は、2つのポイントを使用して、FIFOアドレスのトラックを管理する。1つのポイントは、FIFOに記憶されている第1ワードを指し、もう一方のポイントは、FIFOに記憶されている最後のワードを指し示す。従って、所定のワードへの読み/書き操作を可能にする。読み、又は、書き操作における実施過程において物理的メモリーの終端に到達

した場合には、アドレス発生器は、物理的メモリーのスタートに対して「ラップアラウンド(wraps around)」する。

【0285】要するに、どの符号化規格(MPEGまたはH. 261)が指定されていても、本発明にかかる時間デコーダは、2つの完全なピクチャ又はフレームを記憶することができなければならない。説明を簡易にするために、その中に2つのフレームを記憶しようとするDRAMの物理的メモリーを2つの半分部分に分割するものとし、各半分は、それぞれ、(適切なポイントを用いて)2つのピクチャのうちの特定の1つに対する専用とする。

【0286】MPEGは、3つの異なるタイプのピクチャを用いる、即ち、イントラ(I)、予測(P)、及び双方向補間(B)である。既に述べたように、Bピクチャは、2つのピクチャからの予測に基づく。一方のピクチャは未来から、そして、いま一方は過去から得られる。Iピクチャは、時間デコーダによるそれ以上の復号化を必要としないが、しかし、P及びBピクチャを復号化する際に後で使用するために、2つのピクチャバッファのうちの1つに記憶されなければならない。Pピクチャの復号化には、既に復号化済みのP又はIピクチャから予測を形成することが必要である。復号化されたPピクチャは、P及びBピクチャの復号化に用いるために1つのピクチャに記憶される。Bピクチャは、両方のピクチャバッファからの予測を要求することができる。ただし、Bピクチャは外部DRAMに記憶される。

【0287】I及びPピクチャが復号化される場合に、時間デコーダから出力されないことに注意されたい。その代りに、I及びPピクチャは、ピクチャバッファの1つに記入され、そして、次のIまたはPピクチャが、復号化のために、到着する場合に限り、読み出される。換言すれば、フラッシングに関する本セクションの以降において更に説明されるように、時間デコーダは、2つのピクチャバッファから前のピクチャをフラッシュするために、その次のPまたはIピクチャを信頼する。要するに、空間デコーダは、PまたはIピクチャをフラッシュするために、動画シーケンスの終端において偽のIまたはPを供給することが出来る。結果的に、次の動画シーケンスがスタートするとき、この偽ピクチャはフラッシュされる。Bピクチャの復号化に際して、ピークメモリー帯域幅のロードが起きる。最悪の状態は、全ての予測が半画素の精度を以て作成され、両方のピクチャバッファから供給されるこの種の予測からBフレームが形成される場合である。

【0288】以前に記述したように、時間デコーダは、MPEGピクチャの再順序付けを提供するように、構成することができる。このピクチャ再順序付けにより、データストリーム内のその次のPまたはIピクチャの時間デコーダによる復号化がスタートする時まで、P及びI

ピクチャの出力は遅延する。

【0289】PまたはIピクチャが再順序付けされると、ピクチャがピクチャバッファに記入されるにつれて、特定のトークンは一時的にチップに記憶される。ピクチャがディスプレイ用に読出されると、これらの記憶されているトークンが検索される。時間デコーダの出力において、新規に復号化されたPまたはIピクチャのデータトークンは、より古いPまたはIピクチャと交換される。

【0290】一方、H. 261は、復号化されたばかりのピクチャからのみ予測を製作する。各ピクチャが復号化されるにつれて、2つのピクチャバッファの1つに記入され、次のピクチャ復号化に使用可能となる。必要とされる唯一のDRAMメモリーオペレーションは、8×8ブロックを書くことであり、そして、整数精度のモーションベクトルによって予測を形成することである。

【0291】要するに、動画フォーマット部は、3つのフレーム又はピクチャを記憶する。ピクチャの回復またはスキップするような機能を収容するために、3つのピクチャが記憶される必要がある。

【0292】5. ビットストリームの特性

本発明にかかる空間デコーダを特に参照することとし、符号化されたデータストリームのビットストリーム特性を再検討することは有用である。理由は、これらの特性は、空間デコーダ及び時間デコーダの回路によって扱われなければならないからである。例えば、1つ又は複数の圧縮規格の下において、当該規格の圧縮比率は、1つのピクチャの複数のピクチャを符号化するために使用されるビットの数を減らすことによって達成される。ビット数は、広い範囲に亘って変化し得る。詳細には、このことは、1つのピクチャの基準ピクチャを符号化するために使われるビットストリームの長さは、1つの単位長さとして識別可能であり、他のピクチャは、多数の単位の長さであっても差し支えなく、第3のピクチャが当該単位の部分であることも可能である。

【0293】現行規格(MPEG1、2、JPEG、H. 261)のうちのいずれも、ピクチャ像を終了する方法を定義せず、次のピクチャがスタートする時、現在のピクチャが終わることが黙認されている。更に、規格(特にH. 261)は、エンコーダによって不完全なピクチャが生成されることを可能にする。

【0294】本発明に基づき、トークンの1つピクチャ・エンドを用いることによって1つのピクチャのエンドを示す方法が提供される。ウータートコード検出器から離れる静止符号化データは、ピクチャ・スタートトークンによってスタートし、そして、ピクチャ・エンドトークンによって終了するが、長さが大幅に変化するピクチャから成る。ここに(第1と第2のピクチャとの間)伝送された他の情報があるかもしれないが、第1のピクチャが終了したことは既知である。

【0295】空間デコーダの出力におけるデータストリームは、所定のシーケンスに対して長さ(ビット数)が同じであり、依然としてピクチャスタート、及びピクチャエンドのピクチャから成る。ピクチャスタートとピクチャエンドとの間の時間の長さは変化しても差し支えない。

【0296】動画フォーマット部は、時間的に不均一なこれらのピクチャを受け取り、そして、ドライブされつつあるディスプレイのタイプによって決定される固定ピクチャレートにおいて、これらをスクリーン上にディスプレイする。例えば、PAL-NTSCテレビジョン規格のように種々異なるディスプレイレートが世界中で使用されている。これは、独特の方法において、クチャを選択的にツルーピングまたは反復することによって達成される。普通の「フレームレートコンバータ」、例えば、2-3プルダウンは、固定入力ピクチャレートによって動作するが、動画フォーマット部は可変入力ピクチャレートを扱うことができる。

【0297】6. 再構成可能な処理ステージ

再び図17を参照することとし、再構成可能な処理ステージ(RPS)は、2線インターフェース33、及び入力ラッチ34から入来するトークンを受け取るために用いられるトークンデコード回路32を有する。トークンデコード回路33の出力は、2線インターフェース36及びアクション識別回路39を介して処理回路35に供給される。処理回路35は、アクション識別回路の制御の下におけるデータ処理に適する。処理が完了した後で、処理回路36は、出力ラッチ41を介し完成された信号を2線インターフェースバス40の出力に接続する。

【0298】アクション識別デコード回路39は、2線インターフェースバス40を介してトークンデコード回路33から供給されるか、及び/又は、それぞれ2線インターフェースバス46を介してメモリー回路43及び44から供給される入力を持つ。トークンデコード回路33からのトークンは、アクション識別回路39及び処理回路36に同時に供給される。アクション識別機能並びにRPSについては、本明細書のこの次の部分において表及び図を用いて更に詳細に説明される。

【0299】図17における機能ブロックダイアグラムは、図18、図19、及び図20に記載されている規格から独立した回路ではないステージを示す。データフローは、トークンデコード32を介し、処理ステージ35を介し、そして、出力ラッチ41を介して2線インターフェース回路42に流れる。制御トークンがRPSにより認識される場合には、トークンデコード回路33において復号化され、そして、適切なアクションが行われる。認識されない場合には、出力回路41を介して、2線インターフェース回路42に、変化なしに供給される。現在の発明は、パイプラインを介して制御トークン

の動きを制御するための2線ワイヤインターフェース回路を持つパイプラインプロセッサとして機能する。本発明のこの特徴については、以前に出願された欧州特許出願第92306038、8号に更に詳細に開示済みである。

【0300】本発明において、トークンデコード回路33は、2線インターフェース42を経て現在入来中のトークンがデータトークンまたは制御トークンのいずれであるかを識別するために用いられる。トークンがトークンデコード回路33によって調べられていることが認識された場合には、実施されるべきアクションを示す適切なインデックス信号またはフラグ信号によって、アクション識別回路39に出力される。それと同時に、トークンデコード回路33は、適切なフラグまたはインデックス信号を処理回路に供給し、アクション識別回路39によって扱われつつあるトークンの存在を処理回路に対して警告する。制御トークンも同様に処理されても差し支えない。

【0301】本発明において使用可能な様々なタイプのトークンについては、この後において更に詳細に説明される。本明細書のこの部分に関しては、制御トークンによって運ばれるアドレスはデコーダ33において復号化され、そして、アクション識別回路39内に含まれるレジスタにアクセスするために用いられることに注意するだけで十分である。調査されつつあるトークンが認識された制御トークンである場合には、アクション識別回路39は、ステートマシン全体に亘って制御信号を配分するために、その再構成状態回路を使用する。既に言及したように、これは、アクション識別デコーダのステートマシンを作動化し、このデコーダはデコーダ自体を再構成する。例えば、前記デコーダが符号化規格を変更することもあり得る。このようにして、アクション識別回路39は、図17に示すステートマシンを介して供給しつつある特定の規格を扱うための所要のアクションを復号化する。

【0302】同様に、アクション識別回路39によって制御されている処理回路36は、この事象を引き起こすために適切である場合にデータトークンのデータフィールドに含まれる情報を処理する準備を整える。多くの場合、制御トークンが最初に到達し、アクション識別回路39を再構成し、そして、次に処理回路36によって処理されるデータトークンが直ぐ後に続く。制御トークンは、プロセッシングユニット36内において処理済みのデータトークンの直ぐ前を先行する出力2線インターフェース42を介して出力ラッチ回路41を出る。

【0303】本発明において、アクション識別回路39は、ヒストリステートを保持するステートマシンである。レジスタ43及び44は、トークンデコーダ33から復号化され、そして、これらのレジスタに記憶されている情報を保持する。この種のレジスタは、必要に応じて

て、オンチップ又はオフチップのいずれであっても差し支えない。これら複数のステートレジスタは、アクション識別に接続されたアクション識別回路39において現在識別されつつあるアクション情報を含む。このアクション情報は、以前に復号化されたトークンから記憶され、そして、選定されるアクション影響することが可能である。接続40は、トークンデコード33からアクション識別ブロック39まで直線的に接続する。これは、アクションがトークンデコード回路33によって現在処理されつつあるトークンによっても影響され得ることを示すことを意図したものである。

【0304】本発明に基づくトークン復号化およびデータ処理を示す。データ処理は、アクション識別回路39によって構成されるように行われる。アクションは、条件の数によって影響され、そして、既に復号化済みのトークンから全体的に得られた情報によって影響され、或いは、更に詳細には、レジスタ43及び44において既に復号化済みのトークンから記憶された情報、処理中の現行トークン、及びアクション識別ユニット39がそれ自身で獲得したステートおよびヒストリ情報によって影響される。それによって制御トークンとデータトークンとの間の区別が示される。

【0305】あらゆるRPSにおいて、いくつかのトークンは、当該RPSユニットによって制御トークンとして見られる。この場合、これらのトークンは幾分後の時点においてRPSのオペレーションに影響するはずである。トークンの他の集合は、RPSによってデータトークンとして見られる。この種のデータトークンは、特定回路の設計、既に復号化済みのトークン、及びアクション識別ユニット39の状態によって決定されるような方法においてRPSによって処理される情報を含む。

【0306】特定のRPSは、特定のRPSコントロールに対する或る特定のトークンの集合および他のトークンの集合をデータとして識別するが、これは特定の当該RPSのビューである。別のRPSは、同じトークンに対して別の異なったビューを持つことが可能である。一方においては、いくつかのトークンは、1つのRPSユニットによってデータトークンとして見られ、他方においては、別のRPSユニットによって実際に制御トークンであるものとして決定されることが可能である。例えば、量子化表情報は、少なくともハフマンデコーダ及びステートマシンに関する限りにおいては、データである。理由は、当該情報はその入力にコード化されたデータとして到着し、一連の8ビットワードに書式化され、そして、処理パイプラインを下流に移動する量子化表トークン(QUANTテーブル)と呼ばれるトークンに形作られるからである。当該マシンに関する限りにおいては、全ての情報はデータであったはずであり、当該情報はハンドリングデータであって、或る種のデータを別の種類のデータに変換し、これは明らかに、当該マシンの

当該部分によって遂行される処理機能である。ただし、当該情報が逆量子化器に到着すると、逆量子化器は、複数のレジスタである当該トークン内にその情報を記憶する。実際、64個の8ビット数があり、そして、多数のレジスタであるので、全体として、多数のレジスタが存在可能である。この情報は、制御情報として見なされ、従って、情報は各データワードに乗算する数に影響するので、当該制御情報は、その次のデータトークンにおいて行われる処理に影響する。1つのステージは当該トークンをデータであると見なし、そして、他のステージは当該トークンを制御であると見なした例がある。

【0307】本発明に基づくトークンデータは、殆どの場合、マシンを通過するデータであるものと見なされる。重要な態様の1つは、一般に、トークンデコーダを有する回路の各ステージが特定のトークンの集合を探しており、そして、当該ステージが認識しないトークンは、変化しないままでステージを通過し、そして、パイプラインを下流に移動することである。その結果、現行ステージに後続する下流ステージがこれらのトークンを見ると言う利点があり、これらのトークンに応答可能である。これは、重要な特徴である。すなわち、隣接していないブロックに対してトークンメカニズムを用いることによりブロックの間の通信が可能である。

【0308】本発明の他の重要な特徴は、回路の各々のステージは、各々の規格にとって必要なオペレーションを行うことができる範囲内において、処理能力を持つこと、及び所定の時点において行われなければならないオペレーションに関してトークンとして制御を遂行することである。この能力を提供するために、異なるステージの間において異なる1つの処理エレメントがある。パーザーのステートマシンROMにおいて、3つの完全に異なる分離されたプログラムがある。各プログラムは扱われる各規格に対応する。どのプログラムが実行されるかはコーディング・スタンダードトークンに依存する。換言すれば、これら3つのプログラムの各々は、その内部に、復号化及びコーディング・スタンダード規格トークンの両方を扱う能力もつ。これらのプログラムの各々が、次に解釈されるべき符号化規格がどちらであるかを理解した場合、これらのプログラムは、マイクロコードROM内の当該プログラムのためのスタートアドレスヘリテラルにジャンプする。これが、ステージが多重規格性を扱う方法である。

【0309】異なる規格によって2つの事柄が影響される。第1に、スタートマーカコードの長さを検出するためのシフトレジスタを再構成するためにビットストリーム内のビットのどのパターンが、スタートコード、またはマーカコードとして認識されるかが影響される。第2に、当該スタートまたはマーカコードが何を意味するかを表示する1つの情報がマイクロコード内に所在刷る。ビットの符号化は、3つの規格の間で異なることを

思い出されたい。従って、マイクロコードは、そのコンプレッサ規格に特有の表において、当該規格標準から独立しているもの、即ち、入来コードを表すトークンのタイプを参照する。大抵の場合に、様々な規格の各々は、当該規格が生成する特定のコードを提供するので、このトークンは、一般に、当該規格から独立している。

【0310】逆量子化器79は数学的能力を持つ。量子化器は、乗算および加算を行い、そして、パラメータによって構成される3つ全ての圧縮規格を実行する能力を持つ。例えば、制御装置内のROMにおけるフラグビットは、逆量子化器に対して、定数Kを加えるかどうかを告げる。別のフラグは、逆量子化器に対して、他の定数を加えるかどうかを告げる。逆量子化器は、トークンが量子化器によって流れる場合、レジスタ内にコーディング・スタンダードトークンを記憶する。それ以降にデータトークンが通過する場合には、逆量子化器は、当該規格が何であるかを記憶し、そして、適切なオペレーションを行うために処理エレメントに供給する必要のあるパラメータを探索する。例えば、逆量子化器は、特定の圧縮規格に対してKが0にセットされるか、或いは、1にセットされるかを探索し、そして、その結果をその処理回路に供給する。同様の意味において、ハフマンデコード56は、その中に多数の表を持ち、その幾つかはJPE G用であり、その幾つかはH. 261用であり、その幾つかはMPEG用である。実際、これらの表の大多数は、これらの圧縮規格の1つ以上に役立つ。どの表が使われるかは、当該規格のシンタックスに依存する。ハフマンデコードは、どの表を使うかを告げるステートマシンからのコマンドを受け取ることにより、作動する。従って、ハフマンデコードは、それに入力され、記憶され、そして、どの符号化が実施されつつあるかを表明する1つの状態をそれ自体は直接持たない。そうでなくて、一緒になってそれらの中に情報を含むパーザーステートマシンとハフマンデコードの組み合わせである。

【0311】本発明の空間デコードに関しては、アドレス生成は、改造され、そして、図17に示す場合に類似し、多数の情報は、例えば符号化規格のようなトークンから復号化される。同様に、符号化規格および付加的情報はレジスタに記録され、そして、当該情報がシステム内のマクロブロックを通過して、1つ1つカウントする場合にはアドレス発生器ステートマシンの進行に影響する。最後のステージは、2つのモードH. 261、またはMPEGのどちらかで作動し、そして、容易に識別される予測フィルタ407-9 (図26) である。

【0312】7. 多重規格符号化

更に詳細には、本発明に基づく制御トークンは、当該トークンに複数のワードを有する。この場合、拡張ビットとして知られる1つのビットがセットされ、追加情報を運ぶためのトークン内における付加ワードの使用を指定する。これらの付加制御ビットのうちの或るものは、対

応するステートマシンにおいて1組の規格から独立したインデックス信号を作成するために使用される情報を示すインデックスを含む。トークンの残りの部分は、パイプラインプロセッサを通過する全てのデータストリームに対して標準となる内部の処理制御機能を示し、そして、識別するために使用される。

【0313】本発明の1つの形のにおいて、トークン拡張は、マシン全体に互って配分された相対トークン復号化回路によって復号化される現行符号化規格を運ぶために使われ、そして、そして、新しい符号化規格の下で動作することが適切である場合にはいつでもマシン全体に互るステージのアクション識別回路39を再構成するために使われる。更に、トークン符号化回路は、当該回路が扱うように設計された選定された規格の1つと制御トークンが関係があるかどうかを示すことができる。

【0314】更に詳細には、MPEGスタートコード、及びJPEGマーカにはは8ビットバリューが後続する。H. 261スタートコードのは4ビットバリューが後続する。この文脈において、スタートコード検出器51は、MPEGスタートコードか又はJPEGマーカのいずれかを検出することによって、次の8ビットがスタートコードと関連している値を含むことを示す。次に、前記検出器は、独立的に、MPEGスタートコードまたはJPEGマーカのいずれであり、そして、H. 261スタートコードでないことを示す信号を作成することができる。この第1の場合において、8ビットバリューはデコード回路に入力され、デコード回路の一部は、当該回路を通過するトークンを扱うために現行回路内において使用されるインデックス及びフラグを示す信号を作成する。これは、その後においてどちらの規格が扱われつつあるかを決定するために参照される制御トークンの部分を挿入するためにも使われる。この意味において、制御トークンは、MPEG規格、並びに、同伴データに作用するオペレーションのタイプを示す部分と関係があることを示す部分を含む。既に検討したように、この情報は、その目的のために作られた様々な規格によって必要とされる機能を遂行するために用いられる処理ステージを再構成するために、システムにおいて利用される。

【0315】例えば、H. 261スタートコードに関して、このコードはスタートコードのすぐ後に後続する4ビットバリューと関連する。スタートコード検出器は、この値をトークン発生器ステートマシンに供給する。この値は、3ビットスタート数を作る8ビットデコードに供給される。スタート数は、当該バリューによって示されるように、ピクチャ数のピクチャ・スタートを識別するために用いられる。更に、システムは、既に述べた2線インターフェースの原理の下で作動する多重並列処理パイプラインを含む。各々のステージは、一般に図17に示す形をとるマシンを含む。トークンデコードステー

ジ33は、アクション識別回路39、またはプロセッシングユニット36にステートマシンを適宜エントリさせつつあるトークンを導くために用いられる。プロセッシングユニットは、次の以前の制御トークンによって現行符号化規格を扱うために必要な形に再構成済みであり、現行規格は、処理ステージに現在エントリされつつあり、そして、次のデータトークンによって運ばれる。更に、本発明のこの態様に基づき、処理パイプラインにおける連続したステートマシンは、1つの符号化規格、即ち、H. 261の下で機能することが可能であり、一方、前のステージは、例えばMPEGのような個別規格の下で動作することが可能である。制御トークン及びデータトークンの両方を運ぶために同じ2線インターフェースが使われる。

【0316】更に、本発明のシステムは、固定した数の再構成可能な処理ステージと共に多数の符号化規格を復号化することを要求される制御トークンを用いる。更に詳細には、ピクチャが実際に終了する時点を示すことは重要であるので、ピクチャ・エンド制御トークンが用いられる。従って、多重規格マシンを設計する場合には、どの規格復号化技術を使用するかを示す付加的制御トークンを多重規格パイプライン内に作ることが必要である。この種の制御トークンはピクチャ・エンドトークンである。このピクチャ・エンドトークンは、バッファをフラッシュするように強制し、そして、デコーダを介して現行ピクチャをディスプレイへ押し出すために、現行ピクチャが終了したことを示すために使われる。

【0317】8. 多重規格プロセッシング回路—第2オペレーションモード

すでに述べたスタートコード検出器の形の圧縮規格従属回路は、適切なバスを介して、圧縮規格から独立した回路に適宜相互接続される。規格従属回路は、同じバス及び付加的バスを介して、従属独立組み合わせ回路に接続される。規格から独立した回路は、規格従属独立回路に付加的入力进行供給し、同時に、規格従属独立回路は規格から独立した回路に情報を後方提供する。規格から独立した回路からの情報は、他の適当なバスを介して、出力に供給される。表600は、規格従属スタートコード検出器51への入力に適用される多重規格には各符号化されたビットストリーム内に規格従属的な意味を持つ特定のビットストリームが含まれることを示す。

【0318】9. スタートコード検出器

既に述べたように、本発明に基づくスタートコード検出器は、MPEG、JPEG、及びH. 261ビットトリームを扱い、そして、残りのデコードにとって有意である一連の所有権トークンを前記ストリームから生成することができる。多重規格の復号化が達成される一例として示せば、MPEG (1、及び2) ピクチャ・スタートコード、H. 261ピクチャ・スタートコード、及びJPEGスタートオブスキージ(SOS) マーカは、スタ

ートコード検出器によって等価として扱われ、そして、これら全てが内部ピクチャ・スタートトークンを生成する。同様の方法において、MPEGシーケンススタートコード、及びJPSG 501 (スタートオブイメージ) マーカは、両方共、マシンシーケンススタートトークンを生成する。ただし、H. 261規格は等価スタートコードを持たない。従って、スタートコード検出器は、第1のH. 261ピクチャ・スタートコードに応答して、シーケンススタートトークンを生成する。

【0319】前述のイメージは、いずれも、SCD以外には、直接使用されない。むしろ、例えば、マシンピクチャ・スタートトークンは、ビットストリームに含まれるピクチャ・スタートイメージに等価であると思われる。更に、次の事柄を念頭に置かれたい、即ち、マシンピクチャ・スタート自体は、規格におけるピクチャ・スタートの直接的なイメージではない。むしろ、それは、各々の圧縮コード化規格におけるイメージのオペレーションをエミュレートする規格から独立した復号化機能を提供するために他の制御トークンと組み合わせて使われる制御トークンである。制御トークンによって運ばれた情報に基づく回路の再構成と結合した制御トークンの組み合わせ、及び更に、それぞれのステートマシンのトークン復号化回路部分によって生成されたインデックス及び/又はフラグとの組み合わせは、独特である。典型的な再構成可能なステートマシンについては次に続いて説明される。

【0320】再び、表600を参照することとし、左のカラムには規格イメージのグループの名前が示される。右カラムには、当該規格イメージ内には存在しないか、又は、使用されない規格符号化された信号のエミュレーションに用いられるマシン従属制御トークンが示される。

【0321】表600において、既に述べたように、表600に記載されている規格信号の任意の1つを復号化する場合、マシンシーケンススタート信号はスタートコード検出器によって生成されることが分かる。スタートコード検出器は、システム全体に亘って用いられる2線インターフェースへ供給するために、シーケンススタート、グループスタート、シーケンスエンド、スライススタート、ユーザーデータ、エクストラデータ、及びピクチャ・スタートトークンを作る。これらの制御トークンと共に作動するこれらの各ステージは、トークンの内容によって構成されるか、或は、トークンの内容によって作られるインデックスによって構成され、そして、ピクチャデータトークンが当該ステーションに到着する場合に受信されると予測されるデータを扱う準備が整えられる。

【0322】既に述べたように、例えばH. 261のような圧縮規格の1つは、そのデータストリーム内にシーケンスイメージスタートを持たず、そのデータストリー

ム内にピクチャ・エンドイメージも持たない。スタートコード検出器は、入来ビットストリーム内にピクチャ・エンドポイントを示し、そして、PICTURE Z E E N Dトークンを作る。この点に関して、本発明のシステムは、本発明の実現において使用するために選定された各々のレジスタの位置に情報の1ビットを含むように完全にバックされたデータワードを運ぶことを意図する。この目的のために、2つのスタートコードの間で供給されるビット数として15ビットが選定された。勿論、当該技術分野における通常の熟練者であれば、15ビットより多いか少ないいずれの選択を行うことも可能であることが理解できるはずである。換言すれば、スタートコード検出器からDRAMインターフェースへ供給される全てのデータワードが15ビットであることは、適切なオペレーションのために必要とされる。従って、スタートコード検出器は、データトークンの最終ワード挿入されるパディングと称する余分のビットを作る。説明のためには、15データビットが選定された。

【0323】本発明に基づくパディング(padding)オペレーションを行うためには、多数の2進1が後続する2進0は、15ビットデータワードを完成するために、自動的に挿入される。次に、このデータは、コード化されたデータバッファを介して供給され、そして、パディングを除去するハフマンデコードに供給される。従って、任意の数のビットが、固定したサイズ及び幅のバッファを供給可能である。

【0324】1つの実施例において、ピクチャのスライスを識別するためにスライススタート制御トークンが使われる。スライススタート制御トークンは、ピクチャを更に小さい領域に分割するために用いられる。この領域のサイズは、エンコードによって選定され、そして、スタートコード検出器は、スタートコード検出器から下流に位置するマシン従属ステートステージに対して、受信ピクチャを更に小さい領域に分割するために、スライススタートコードのこの唯一のパターンを識別する。領域のサイズは、スタートコード検出器で識別され、そして、符号化されたピクチャを伸長するために再組合せ回路及び制御トークンによって使用されるエンコードによって選択される。スライススタートコードは、主としてエラー回復のために使われる。

【0325】スタートコードは、デコードを始める唯一の方法を提供し、そして、これについては、続いて更に詳細に説明される。スタートコード検出器をコード化されたデータバッファの前に配置すれば、コード化データバッファの後、そして、ハフマンデコード及び動画dママルチプレクサ前にスタートコード検出器を配置する場合と比較して、多数の利点が得られる。第1のバッファの前にスタートコード検出器を設置した場合には、1) トークンの組立てが可能であり、2) 例えばスタートコードのような規格制御信号の復号化が可能であり、3)

データがバッファに入る前に、ビットストリームのパッドが可能であり、そして、4) バッファを空にするために、制御トークンの適切なシーケンスを作り、利用可能なデータをバッファからハフマンデコードに入れることが可能である。

【0326】スタートコード検出器によって出力される制御トークンの殆どは、様々なピクチャ、及び動画符号化規格の構文エレメントを直接反映する。スタートコード検出器は、構文エレメントを制御トークンに変換する。これらの自然トークンに加えて、幾つかの一意的、及び/又は、マシン従属トークンが生成される。一意的なトークンには、本発明のシステムにより使用するために特に設計され、内部的にも外部的にも一意的であり、そして、本発明の多重規格性に関して補助するために用いられるトークンが含まれる。この種の一意的なトークンの例には、ピクチャ・エンド、及びコーディング・スタンダードが含まれる。

【0327】更に、トークンは、符号化規格の間の構文的な差の幾つかを除去するため、及びエラー条件と協調動作するために導入される。自動トークン生成は、規格従属データの順次分析の後で行われる。従って、空間デコードは、空間デコードの入力に直接供給されたトークン、即ち、SCD、並びに、コード化されたデータにおけるスタートコードの検出に続いて生成されたトークンに平等に応答する。本発明の多重規格性を制御するために一連のエキストラトークンが2線インターフェースに挿入される。

【0328】MPEG及びH. 261符号化動画ストリームは、規格従属非データ識別可能ビットパターンを含み、これらのパターンの1つを、今後、スタートイメージ、及び/又は、規格従属コードと呼ぶこととする。JPEGにおいては、同様の機能はマーカコードによって提供される。これらのスタートする/マーカコードは、コード化されたデータストリームの構文の重要な部分を識別する。スタートコード検出器によって行われるスタート/マーカコードの分析は、コード化されたデータ解析の最初のステージである。

【0329】スタート/マーカコードパターンは、ビットストリーム全体を復号化することなしにパターンを識別できるように設計される。従って、それらのパターンは、エラー回復及びデコードスタートを援助するために本発明に基づいて使用できる。スタートコード検出器は、コード化されたデータ構成内のエラーを検出し、そして、デコードのスタートを援助するための、機能を提供する。スタートコード検出器のエラー検出能力については、デコードのスタート過程につれて、この後で詳細に説明される。

【0330】前記の記述は、主として、マシン従属ビットストリームの特性、及び本発明のアドレッシング特性との関係に関する説明であった。次の記述は、スタート

コード検出器に関する規格従属コード化データのビットストリームの特性の説明である。

【0331】各々の規格圧縮符号化システムは、特定の圧縮仕様を識別するために選定された独特のスタートコードコンフィギュレーション、またはイメージを用いる。各々のスタートコードは、スタートコードバリューを持つ。スタートコードバリューは、当該規格の言語内においてスタートコードと関連するオペレーションのタイプを識別するために用いられる。本発明の多重規格デコードにおいては、既に述べたように、その互換性は、制御トークン及びデータトークンコンフィギュレーションに基づく。フラグ信号を含むインデックス信号は、各ステートマシン内において回路生成され、そして、以下に適宜説明することとする。

【0332】当該規格に含まれるスタート、及び/又は、マーカコード、並びに、データワードと対照的な他の規格ワードは、当該マシンにおいて使用される制御、及び/又は、データトークンの内容に関して、コード、及び/又は、マシン従属コードの使用との混乱を回避するために、イメージとして識別されることが少なくない。同様に、スタートコードと言う用語は、JPEGマーカコード並びにMPEG及びH. 261に関する包括的な用語として用いられることが多い。マーカコードとスタートは同じ目的に役立つ。更に、「フラッシュ」(flush)と言う用語は、使フラッシュトークンを意味する場合と、そして、動詞として、例えば、スタートコード検出器シフトレジスタをフラッシュする場合(「フラッシュ済み」信号を含む)のように、両方の意味に用いられる。混乱を回避するために、フラッシュトークンは必ず大文字で書かれる。用語(動詞、または名詞)の他の全ての使用は小文字による。

【0333】規格従属コード化済み入力ピクチャ入力ストリームは、長さの変化するデータ及びスタートイメージを含む。スタートイメージは、規格に応じて即座に後続する当該データに対してどのオペレーションが実施されるべきかを告げる値と一緒に運ぶ。ただし、多重規格に対して互換性が必要とされる本発明の多重規格パイプライン処理システムにおいて、システムは、全ての規格において全ての機能を扱うために最適化されている。従って、多くの条件において、一意的なスタート制御トークンは、符号化された信号規格イメージの値に含まれる値に関してのみ互換性があるばかりでなく、当該技術分野において周知のように各規格に対して指定されたパラメータによる表示に従って、規格のオペレーションをエミュレートするために各種ステージを制御可能でなくてはならない。この種の規格は、全て、参照資料として本明細書に組み込み済みである。

【0334】単独で、或いは、他の制御トークンと組み合わせることによって規格ビットストリームに含まれる非データ情報をエミュレートするトークンの間の関係を埋

解することが重要である。フラグ信号を含むインデックス信号の個別の集合は当該ステートマシン内におけるいくつかの処理を扱うために各ステートマシンによって生成される。規格に運ばれる値は、規格データ及び非データ信号の扱いをエミュレートするようにマシンの従属制御信号にアクセスするために使用できる。例えば、スライススタートトークンは2ワードトークンであり、従って、既に述べたように、2線インターフェースに入力される。

【0335】本発明のシステムへのデータ入力、例えば、ディスク、テープ等のような、スタートコード検出器51(図18)内における第1機能ステージに8ビットのデータを供給するあらゆる適当なデータソースからのデータソースであっても差し支えない。スタートコード検出器は3つのシフトレジスタを有する、即ち、第1のシフトレジスタは、8ビット幅であり、その次は24ビット幅であり、そして、その次は15ビット幅である。各々のレジスタは、2線インターフェースの一部である。データソースからのデータは、1つのタイミングサイクル期間中に、1つの単一8ビットバイトとして第1レジスタにロードされる。その後で、第1シフトレジスタの内容は、一時に1ビットずつ、デコード(第2)シフトレジスタにシフトされる。24サイクル後に、24ビットレジスタは満杯になる。8サイクル毎に、8ビットバイトが、第1シフトレジスタにロードされる。各バイトは、バリュースhiftレジスタ221(図29)にロードされ、そして、付加的な8サイクルは、バリュースhiftレジスタを空にし、そして、シフトレジスタ231をロードするために用いられる。シフトレジスタ231を空にするためには8サイクルが用いられ、従って、これらの3オペレーション、または24サイクルの後で、24ビットレジスタ内にはまだ3バイトがある。バリュースhiftレジスタ230はまだ空である。

【0336】今24ビットシフトレジスタ内に1つのピクチャ・スタートワードがあるものと仮定すれば、検出サイクルは、ピクチャスタートコードパターンを認識し、そして、その出力としてスタート信号を供給する。一度、検出器がスタートを検出すれば、それに続くバイトは、当該スタートコードと関連した値であり、そして、これは、現在バリュースhiftレジスタ221にシッティング(sitting)中である。

【0337】検出シフトレジスタの内容はスタートコードであると識別済みであるので、これ以上プロセッシングが起きないことを保証するために、これらの3バイトを用いて、2線インターフェースから、この内容を除去しなければならない。デコードレジスタは空にされ、そして、バリュースhiftレジスタ230は、当該値がこの種レジスタまでシフトされるのを待つ。

【0338】この段階において、バリュースhiftレジスタの低位ビット位置の内容は、ピクチャ・スタ

ートと関連した値を含む。標準ピクチャ・スタート信号と等価の空間デコードは、SDピクチャ・スタート信号と呼ばれる。ここにおいて、SDピクチャ・スタート信号自体はトークンヘッダに含まれようとしており、そして、この値は、トークンヘッダに対する拡張ワードに含まれようとしている。

【0339】10. トークン

本発明の実現に際して、トークンは、制御、及び/又は、データ機能に対する対話型インターフェースメッセージパッケージの形式の万能順応ユニットであり、そして、1つの認識済みトークンに回答して様々なオペレーションを行うためにそれ自体を再構成する1つのステージである再構成可能なプロセッシングステージ(RPS)と共に使用するために改作される。トークンは、様々な機能を達成するために処理ステージに応じて位置従属又は位置独立のいずれかであっても差し支えない。更に、トークンは、処理ステージによって変化させられ、その次に、更に別の機能を達成するためにパイプラインを下流方向に供給されるので、変質性であっても差し支えない。トークンは、ステージの全て又は全てより少ない部分と対話し、そして、この観点からすれば、隣接、及び/又は、非隣接のステージと対話するた見なしでも差し支えない。トークンは、幾つかの機能に対しては位置従属であり、そして、他の機能に対しては位置独立であっても差し支えなく、そして、或るステージとの特定の対話は、当該ステージの以前における処理ヒストリによって調整可能である。

【0340】ピクチャ・エンドトークンは、多重規格デコードにおいてピクチャエンドを送信する方法である。

【0341】多重規格トークンは、規格従属及び規格独立ハードウェアと制御トークンとの混合体を用いてMP EG、J P E G、及びH. 261データストリームを1つの単一デコードにマッピングする方法である。

【0342】サーチモードトークンは、ランダムアクセス、及び強化されたエラー回復を可能にするM P E G、J P E G、及びH. 261データストリームを検索するための技術である。

【0343】ストップ・アフタ・ピクチャトークンは、復号化のクリアエンドを達成する方法であり、ピクチャのエンドを送信し、そして、デコードパイプライン、即ち、チャンネルの変更をクリアする。

【0344】更に、トークンをパッドすることは、固定サイズ固定幅のバッファを通して任意の数のビットを供給させる方法である。

【0345】本発明は、トークンおよび2線システムを使用する可変構成を有するパイプライン処理システムに向けられる。2線システムと組合わせた制御トークン及びデータトークンの使用は、制御トークンを使わないシステムと比較して拡張された作動能力を有することを可能にする多重規格システムの達成を容易にする。

【0346】制御トークンは、デコーダプロセッサの中の回路によって生成され、そして、取り扱うために直列パイプラインプロセッサ内に供給する多数の異なるタイプの規格従属信号のオペレーションをエミュレートする。直列プロセッサによる処理及び次に示す項目に関する考察を行うために選定される多重規格の全てのパラメータを検討する技術が用いられる。即ち、1)それらの類似性、2)それらの相違、3)それらの必要性、及び必要条件、及び4)直列プロセッサに送込まれる全ての規格信号を効果的に処理するための正しいトークン機能の選定。トークンの機能は規格をエミュレートすることである。制御トークン機能は、部分的には規格従属信号間のエミュレーション/翻訳として、及びパイプラインプロセッサを介して制御情報を伝えるためのエレメントとして使われる。

【0347】先行該技術によるシステムにおいては、規格を識別し、そして、マイクロプロセッサインターフェースを用いることによって専用回路を作るために周知の技術に従って専用マシンが設計される。マイクロプロセッサからの信号は、専用下流コンポーネントを介してデータの流れを制御するために用いられる。この伸長機能の選択、タイミング、及び組織化は、マイクロプロセッサから入来する信号によって援助された固定論理回路による制御の下に行われる。

【0348】これとは対照的に、本発明のシステムは、制御トークンの制御の下に下流機能ステージを構成する。MPUからの必要な、及び/又は、代替制御を入手するためにオプションが提供される。

【0349】トークンは、伸長回路パイプラインプロセッサを介して情報を伝達するための優れたフォーマットを提供し、そして、作成する。以下に示す好ましい実施例において選定され、そして、使用される設計においては、トークンの各ワードの幅は最小限8ビットであり、そして、1つの単一トークンは、1つ又は複数のワードを越えて拡張することが出来る。トークンの幅は、可変であり、そして、ビット数は任意に選定することが出来る。拡張ビットは、トークンが現行ワードを越えて拡張されるかどうか、即ち、トークンの最終ワード以外のトークンの全てのワードにおけるビットが2進1にセットされているかどうかを示す。トークンの第1ワードがゼロの拡張ビットを持つ場合には、これは、トークンの長さは僅かに1ワードであることを示す。

【0350】各トークンは、トークンの第1ワードのビット7においてスタートするアドレスフィールドによって識別される。アドレスフィールドは、長さが可変であって、そして、複数のワードを越えて拡張することが潜在的に可能である。好ましい実施例において、アドレスの長さは8ビット以内である。ただし、これは、本発明における制限条件でなく、これらのトークンの使用によって達成されるように選定された処理過程の大きさを意

味する。ワード1及び2における拡張ビットが1つの1であり、付加ワードの後続を意味することは拡張ビット識別ラベルによって示されることに注意されたい。ワード3における拡張ビットはゼロであり、従って、当該トークンのエンドを示す。

【0351】更に、トークンは可変ビット長さである。例えば、9ビットのトークンワードに拡張ビットを加えて合計10ビットが用いられる。本発明の設計において、出力バスの幅は可変である。空間デコーダからの出力の幅は9ビットであるか、または拡張ビット含まれる場合には10ビット出ある。好ましい実施例において、これらの余分なビットを利用する唯一のトークンはデータトークンであり、全ての他のトークンは、このエキストラビットを無視する。これは、制限条件ではなく、単に実施例に過ぎないことを理解されたい。

【0352】データトークン及び制御トークンコンフィギュレーションを使用することにより、これらのデータトークンによって運ばれつつある1つのワードにおけるビットの数によって表されるデータの長さを変えることが可能である。例えば、データトークンのワードにおけるデータビットは、この直列伸長プロセッサ全体を通じて使用される等速呼出記憶装置にアクセスする際に使用するために、同一データトークンの他のワードにおけるデータビットと組み合わせて11ビット、或いは、10ビットのアドレスを形成可能であることは検討済みである。これによって、広い範囲に互り汎用性を持つことを容易にし、可変性が追加される。

【0353】既に述べたように、データトークンは、1つの処理ステージから次の処理ステージにデータを運ぶ。従って、このトークンの特性は、デコーダを通過する際に変化する。例えば、空間デコーダへの入力において、データトークンは、8ビットワードにバックされたビット直列符号化動画データを運ぶ。ここでは、各トークンの長さには制限がない。ただし、本発明のこの態様の汎用性を例示するために(空間デコーダ回路の出力において)、各データトークンは、ちょうど64ワードを持ち、そして、各ワードは9ビット幅であるものとする。更に詳細には、規格符号化信号は、長さの異なるメッセージに対して、画像の異なる強度および詳細をコード化することを可能にする。グループの第1画像は、プロセッシングユニット最多情報を提供するので、通常、最も長い数のデータビットを持つ。結果として、第1画像は、できる限り多くの情報の伸長をスタートすることが出来る。後続するワードは、走査情報フィールドにおける第2の位置に関して第1ワードと比較した場合に違った信号を含むので、一般に、長さが短い。規格符号化システムの要請に従い、可変量のデータが空間デコーダの入力に供給されるように、ワードは、相互に散在させられる。ただし、空間デコーダが機能した後においては、情報は、スクリーン上のディスプレイに適し

たピクチャフォーマットレートにおいてその出力に供給される。例えば、NTSC、PAL、及びSECAMのような様々なディスプレイシステムと全世界に亘ってインターフェイスするために空間デコーダの時間に関する出力レートは変化可能である。動画フォーマッピング部は、この可変ピクチャレートをディスプレイに適した一定のピクチャレートに変換する。ただし、ピクチャデータは、依然として、64ワードから成るデータトークンによって運ばれる。

【0354】11. DRAMインターフェース

3個のデコーダチップの各々に1つの単一高性能構成可能DRAMインターフェースが用いられる。一般に、各チップ上のDRAMインターフェースは実質的に同じである。ただし、インターフェースは、チャンネル優先権を扱う方法が、相互に異なる。このインターフェースは、空間デコーダ、時間デコーダ、及び動画フォーマッピング部によって使用される外部DRAMを直接ドライブするように設計されている。一般に、DRAMインターフェースをこれらのシステムのDRAMに接続するには、外部のロジック、バッファ、またはコンポーネントは必要でない。

【0355】本発明に基づき、インターフェースは、次の2つの方法により構成可能である。1. 種々異なるDRAMタイプを収容するように、詳細なタイミングのインターフェースを構成することが出来る。

【0356】2. 異なる用途に応じてコスト/性能トレードオフを提供するように、DRAMへのデータインターフェースの幅を構成することが出来る。

【0357】一般に、DRAMインターフェースは、システム内の各々3つのチップ上で実現される規格から独立したブロックである。再び、これらのチップは、空間デコーダ、時間デコーダ、及び動画フォーマッピング部である。図18、図19、図20を再度参照することとし、これらの図は、DRAMインターフェースと、それぞれ、空間デコーダ、時間デコーダ、及び動画フォーマッピング部の残りのブロックとの間の関係を表すブロック図である。各チップにおいて、DRAMインターフェースは外部DRAMにチップを接続する。現在では、必要とされる比較的大量のDRAMをチップ上に作成することは実際のでないで、外部DRAMが使用される。(注記：各チップは、自身の外部DRAM及び自身のDRAMインターフェースを持つ。)更に、DRAMインターフェースが圧縮規格独立である場合には、多重規格H. 261、JPEG、及びMPEGの各々を実現するように、インターフェースを構成しなければならない。多重規格オペレーション用としてのDRAMインターフェースをどのように再構成するかについては、後で更に詳細に説明される。

【0358】DRAMインターフェースのオペレーションを理解するには、DRAMインターフェースとアドレ

ス発生器との間の関係、及び2線インターフェースを用いてこれら両者の間でどのように交信するかということを理解することが必要である。

【0359】一般に、名前が意味するように、アドレス発生器は、DRAMをアドレスするために(例えば、DRAM内の特定アドレスに対して読み書きを行う場合)DRAMインターフェースが必要とするアドレスを生成する。2線インターフェースを使用する場合、DRAMインターフェースが(パイプラインにおける先行ステータスから)双方のデータ、及び有効なアドレス(アドレス発生器から)を持つ場合に限り読むこと、及び書くことが起きる。個別のアドレス発生器の使用は、後で更に検討するように、アドレス発生器及びDRAMインターフェース両方の構成を簡素化する。

【0360】本発明において、DRAMインターフェースは、アドレス発生器、および、それを介してデータが供給されるステージのクロックの両方に同期するクロックから操作可能である。操作の非同期性を扱うためには特殊な技術が必要とされている。

【0361】データは、一般に、DRAMインターフェースと64バイトのブロックにおけるチップの残りの部分との間に転送される(時間デコーダにおける予測データだけは唯一の例外である)。転送は、「スイングバッファ」として知られているデバイスによって行われる。スイングバッファは、本質的に、1つのRAMを満たすか、或いは、空にし、同時に一方ではチップの他の部品がもう一方のラムを空にするか、或いは、満たすDRAMインターフェースを用いてダブルバッファされたコンフィギュレーションにおいて操作される1対のRAMである。アドレス発生器からのアドレスを運ぶ個別のバスは各スイングバッファと関連する。

【0362】本発明において、各々のチップは、4つのスイングバッファを持つが、これらのスイングバッファの機能は各場合毎に異なる。空間デコーダにおいては、1つのスイングバッファは、コード化されたデータをDRAMに転送するために使われ、も1つのスイングバッファはDRAMからコード化されたデータを読むために用いられ、第3のスイングバッファは、トークン化されたデータをDRAMへ転送するために使用され、そして、第4のスイングバッファはトークン化されたデータをDRAMから読み出すために用いられる。ただし、時間デコーダにおいては、1つのスイングバッファは、イントラ又は予測ピクチャデータをDRAMに書き込むために使われ、第2のスイングバッファはイントラ又は予測データをDRAMから読み出すために用いられ、そして、他の2つは、前方、及び後方への予測データを読み取るために使われる。動画フォーマッピング部においては、1つのスイングバッファは、データをDRAMへ転送するために使われ、そして、他の3つは、DRAMからデータを読むために使われ、輝度(Y)及び赤および青

の色差データ(各々、Cr、及びCb)用として1つずつ使用される。1つの書き込みスイングバッファ、及び1つの読み取りスイングバッファを有する1つの仮想DRAMインターフェースのオペレーションについて、次のセクションにおいて説明することとする。本質的に、これは、空間デコーダのDRAMインターフェースのオペレーションと同じである。オペレーションについては図32に示す。

【0363】図32は、アドレス発生器301、DRAMインターフェース302、及びデータを供給するチップの残りのステージの間の制御インターフェースは全て2線インターフェースであることを示す。アドレス発生器301は、制御トークンを受信する結果としてアドレスを生成するか、或いは、単にアドレスの固定したシーケンスを生成(例えば、空間デコーダのFIFOバッファ用)しても差し支えない。DRAMインターフェースは、アドレス発生器301と関連している2線インターフェースを特別な方法で扱う。アクセプトラインがアドレスを受信する準備が整った場合にアクセプトをハイに保持する代わりに、アドレスラインは、アドレス発生器による有効なアドレス供給を待ち、当該アドレスを処理し、そして、1クロック周期に亘ってアクセプトラインをハイにセットする。このように、リクエスト/肯定応答(PEQ/ACK)プロトコルが実現される。

【0364】DRAMインターフェース302の独特の特徴は、アドレス発生器301、及びデータを供給、或いは、受け入れるステージと独立して通信することのできる能力である。例えば、アドレス発生器は、書き込みスイングバッファ(図33)におけるデータと関連しているアドレスを生成しても差し支えないが、しかし、書き込みスイングバッファが外部DRAMへの記入準備が整ったデータのブロックが存在することを送信する時までアクションは行われない。同様に、書き込みスイングバッファは、外部DRAMへの記入準備が整っているもののデータのブロックを含んでも差し支えないが、アドレスが、アドレス発生器301から当該バスに供給される時まで、アクションは行われない。更に、書き込みスイングバッファにおけるRAMの1つが一度データで満たされると、データ入力が停動される(2線インターフェース受け入れ信号がローにセットされる)以前に、他のRAMは完全に満たされ、そして、DRAMインターフェース側に「スイング」されることもあり得る。

【0365】本発明にかかるDRAMインターフェイス302のオペレーションを理解する際、適切に構成されたシステムにおいて、DRAMインターフェースは、スイングバッファと外部DRAM303との間において、スイングバッファとチップの残りの部分との間の全ての平均データレート合計と少なくとも同等のデータレートを以てデータ転送が可能であることに注意することが重要である。

【0366】各DRAMインターフェース302は、どちらのスイングバッファが次にサービスするかを決定する。一般に、これは、「ラウンドロビン」(round robin)(即ち、次にサービスを受けるスイングバッファは、最も最近でない順番であった次の利用可能なスイングバッファである)、或いは、優先位エンコーダ(即ち、この場合、幾つかのスイングバッファが他のものより更に高い優先位を持つ)のいずれかである。双方の場合に、他の全てのリクエストより更に高い優先位を持つ追加リクエストはリフレッシュリクエスト発生器から入来する。リフレッシュリクエストは、マイクロプロセッサインターフェースを介してプログラム可能なリフレッシュカウンタから生成される。

【0367】ここで、図33を参照することとし、この図には、書き込みスイングバッファの構成図が示される。書き込みスイングバッファインターフェースには、第1RAM311、及び第2RAM312の2つのブロックが含まれる。ここで更に検討するように、データは、書き込みアドレス303、及び制御部314による制御の下で、前のステージからRAM311及びRAM312に記入される。RAM311、及びRAM312から、データは、DRAM315に記入される。DRAMへのデータ記入中は、ここで更に説明されるように、DRAM行アドレスは、アドレス発生器によって供給され、そして、列アドレスは、書き込みアドレス及び制御部によって供給される。オペレーションにおいて、有効なデータは、入力316に供給される(データイン)。一般に、データは、前のステージから受け取られる。各々のデータは、DRAMインターフェースによって受け入れられるにつれて、RAM311に記入され、そして、書き込みアドレス制御部は、RAM311をインクリメントし、次の1つのデータのRAM311への記入を可能にする。データのRAM311への記入は、それ以上データが無い、或いは、RAM311が満杯になるまで、継続される。RAM311が一杯である場合には、入力側は、制御を断念し、そして、RAM311の読み取り準備が整っていることを示すために、信号を読み取り側に送る。従って、この信号は、2つの非同期クロックレームの間を供給し、3つの同期化フリップフロップを介して供給する。RAM312が空であるものと仮定すれば、入力側に到着するデータの次の項目はRAM312に記入される。そうでない場合には、RAM312が空になった場合にこれが起きる。ラウンドロビン又は優先位エンコーダが(特定のチップによってどちらが使用されるか応じて)読み取られる順番に当たるスイングバッファを指示した場合、DRAMインターフェースは、RAM311の内容を読み、そして、それらを外部DRAMに記入する。次に、信号は、非同期インターフェースを通して後方に送られ、RAM311が再度満杯にされる準備が整っていることを示す。

【0368】DRAMインターフェースがRAM311を空にし、そして、入力側がRAM312を満たす以前に、それを「スイング」する場合には、データのスイングバッファによる連続受け入れが可能である。そうでない場合には、RAM2が満杯である場合には、スイングバッファは、RAM311が、入力側による使用のために後方に「スイング」されるまで、その受け入信号をローにセットする。

【0369】本発明に基づく読み取りスイングバッファのオペレーションは、入力データと出力データが反転する点を除けば、同じである。

【0370】本発明のDRAMインターフェースは、利用可能なメモリーバンド幅を最大限にするように設計されている。データの各8×8ブロックは、同一DRAMページに記憶される。このようにして、DRAM高速ページアクセスモードを完全に利用できる。この場合、1つの行アドレスが供給され、多くの列アドレスがこれに続く。特に、行アドレスは、アドレス発生器によって供給されるが、列アドレスは、後で検討するように、DRAMインターフェースによって供給される。更に、外部DRAMへのデータバスの幅が8、16、または32ビットであることを可能にする機能が提供される。従って、DRAMの使用量は、特定用途のサイズ及びバンド幅必要条件にマッチさせることができる。

【0371】この例において（正確には、空間デコーダ上のDRAMインターフェースがどのように作動するかと言うこと）、アドレス発生器は、読み取り、及び書き込みスイングバッファの各々に対するブロックアドレスをDRAMインターフェースに供給する。このアドレスは、DRAMのための行アドレスとして使われる。6ビットの列アドレスは、DRAMインターフェース自体によって供給され、そして、これらのビットは、スイングバッファラムのためのアドレスとして同様に使われる。スイングバッファへのデータバスの幅は32ビットである。従って、外部DRAMへのバス幅が32ビット未満である場合には、2或いは4の外部のDRAMアクセスは、書き込みスイングバッファから次のワードが読み取られる以前、或いは、次のワードが読み取りスイングバッファに記入される以前に、行われなければならない（読み取り、及び書き込みは、外部DRAMに対する転送の方向を意味する）。

【0372】時間デコーダ、及び動画フォーマット部の場合においては、条件は更に複雑である。時間デコーダのアドレッシングは、このセクションにおいて更に検討するようにその予測的態様のために、更に複雑である。動画フォーマット部に関して本セクションにおいて更に検討されるように、多重動画出力規格態様のために、動画フォーマット部のアドレッシングは更に複雑である。

【0373】既に述べたように、時間デコーダは4つの

スイングバッファを持つ。その中の2つは復号化済みイント及び予測（Iお呼びP）ピクチャデータの読み取り及び書き込みを実行するために用いられる。これらは、既に説明したように作動する。もう一方の2つは、予測データを受け取るために使われる。これらのバッファは更に興味深い。

【0374】一般に、予測データは、運動モーションベクトルにx及びyとして指定されているように、処理済みブロックの位置から偏る。従って、検索されるべきデータブロックは、符号化された（そしてDRAMに書き込まれた）ので、一般に、当該データのブロック境界と一致しない。これは、図34に示すように、陰を付けた部分は形成されつつあるブロックを表し、点線で示すアウトラインは、それに基づいて予測がなされつつあるブロックを表す。アドレス発生器は、モーションベクトルによって指定されたアドレスを、大きい矢印で示すように、オフセットされたブロック（ブロックの全数）に変換し、また、小さな矢印によって示すようにオフセットされたピクセルに変換する。

【0375】アドレス発生器において、DRAMから検索されるべきブロックのアドレスを形成するために、フレームポインタ、ベースブロックアドレス、及びベクトルオフセットが加えられる。ピクセルオフセットがゼロである場合には、ただ1つのリクエストが生成される。x又はy次元のいずれかにオフセットがある場合には、2つのリクエスト、即ち、もとのブロックアドレス及び直ぐ下のブロックアドレスが生成される。x及びyにオフセットのある場合には、4つのリクエストが生成される。検索されるべき各々のブロックに関して、図の例に最もよく示されているように、アドレス発生器は、スタート、及びストップアドレスを計算する。

【0376】図35の陰の部分に相当するピクセルオフセット（1，1）について考察することとする。アドレス発生器は、図においてAからDまでにラベル表示される4つのリクエストを作る。解決すべき問題は、行アドレスの所要のシーケンスをいかに高速に供給するかと言うことである。解答は、次に示すように「スタート/ストップ」技術を使用することである。

【0377】図35のブロックAについて考察することとする。読み取りは、位置（7，7）においてスタートし、位置（1，1）において終了しなければならない。現段階において、一時に1バイトが読み取られつつあるものと仮定する（即ち、8ビットDRAMインターフェース）。座標対におけるx値は、アドレスの3つのLSBを形成し、y値は3つのMSBを形成する。x及びyスタートバリューは双方共に1であり、アドレスに9を提供する。データはこのアドレスから読み取られ、そして、x値はインクリメントされる。プロセスは、x値がそのストップバリューに到達する時まで繰り返され、到達した時点において、yバリューは1だけインクリメント

され、そして、xスタートバリューは再ロードされ、1つのアドレス17を与える。データの各バイトが読み取られるにつれて、xバリューは、ストップバリューに到着するまで、再びインクリメントされる。x及びyの両バリューがそれらのストップバリューに到達するまで、プロセスは繰り返される。このようにして、9、10、11、12、13、14、15、17...23、25、...、31、33、...、57、...、63のアドレスシーケンスが生成される。

【0378】同様の方法において、ブロックBに対する10スタート及びストップ座標は(1, 0)及び(7, 0)であり、ブロックCに対しては(0, 1)及び(0, 7)であり、そして、ブロックDに対しては(0, 0)及び(0, 0)である。

【0379】次の問題は、このデータをどこに記入するかである。明らかに、ブロックAを観察すれば、アドレス9から読み取られたデータは、スイングバッファのアドレス0に記入されなければならない、アドレス10からのデータは、スイングバッファのアドレス1に記入されなければならない。等々。同様に、ブロックBにおける20アドレス8から読み取られたデータは、スイングバッファにおけるアドレス15に記入されなければならない、そして、アドレス16からのデータは、スイングバッファにおけるアドレス15に記入されなければならない。この機能は、次に概説するように、非常に簡単に実現できる。

【0380】ブロックAについて考察するものとする。読取り開始に際して、スイングバッファアドレスレジスタは、ストップバリューの逆値がロードされる。yの逆ストップバリューは3つのMSBを形成し、そして、x30の逆ストップバリューは3つのLSBを形成する。この場合、DRAMインターフェースは外部DRAMにおけるアドレス9を読みつつあるが、スイングバッファアドレスはゼロである。表500「予測アドレッシング」に示すように、外部DRAMアドレスレジスタがインクリメントされるにつれて、スイングバッファアドレスレジスタもインクリメントされる。

【0381】ここまでの説明は、8ビットDRAMインターフェースに集中してきた。16または32ビットインターフェースの場合には、2、3のささいな修正が行われなければならない。最初に、ピクセルオフセットベクトルは、それが16または32ビット境界を指すように「クリップ」しなければならない。ここで用いてきた例において、ブロックAに関しては、読み取られた第1のDRAMはアドレス0を指し、そして、アドレス0から3までのデータが読み取られる。2番目に、不要データは放棄されなければならない。これは、データ全てをスイングバッファに書き込み(この場合、スイングバッファは、8ビットの場合に必要なよりも物理的に差に大きくなければならない)、そして、オフセットと50

共に読み取ることにより遂行される。MPEG半画素補間を実施する場合には、x及び/又はyにおける9バイトが、DRAMインターフェースから読み取られなければならない。この場合、アドレス発生器は、適切なスタート及びストップアドレスを提供する。DRAMインターフェースにおいてはいくらかの付加ロジックが用いられるが、しかし、DRAMインターフェースが作動する方法には基本的な変化はない。本発明の時間デコーダDRAMインターフェースに関して注意すべき最後の点は、データに関してどのような処理を実施することが必要とされるかを示すために、予測フィルタに付加情報が提供されなければならないことである。必要な処理の内容を次に示す。

【0382】転送(64、72、又は、81バイトの)の最後のバイトを示す「最終バイト信号」。

【0383】H. 261フラグ。

【0384】双方向性予測フラグ。

【0385】ブロックの次元(x及びyにおける8又は9バイト)を示す2つのビット。

【0386】ブロックの順序を示す1つの2ビット番号。

【0387】スイングバッファからデータが読み出されるにつれて、最終バイトフラグが生成可能である。他の信号は、アドレス発生器から得られ、そして、予測フィルタブロックによってデータがスイングバッファから読み出されるにつれて、信号がデータの正しいブロックと組合わされるように、DRAMインターフェースを経てパイプに通される。

【0388】動画フォーマッティング部において、データは、ブロックにおける外部DRAMに記入されるが、しかし、ラスト順に読み出される。書き込みは、空間デコーダに関して既に説明した場合と全く同じであるが、しかし、読取りは少しだけ更に複雑である。

【0389】動画フォーマット部、外部DRAMにおけるデータは、少なくとも8つのブロックのデータが1つの単一ページに適合するように組織される。これら8つのブロックは8つの連続した水平ブロックである。ラスト化に際して、8つの連続したブロックの各々から8つのバイトが読み出され、そして、スイングバッファに記入されることが必要である(即ち、8ブロックの各々における同じ列)。

【0390】最上列について考察することとし(そして、1バイト幅のインターフェースを仮定する)、yアドレス(3MSBS)の場合と同様にxアドレス(3LSBS)はゼロにセットされる。次に、最初の8バイトが各々読みとられるにつれて、xアドレスはインクリメントされる。この時点において、アドレスの上位部分(ビット6及びそれ以上—LSB=ビット0)のトップ部品はインクリメントされ、xアドレス(8LSBS)はゼロにリセットされる。64バイトが読み出されるま

で、このプロセスは繰り返される。外部DRAMへのインターフェース幅が16または32ビットである場合には、xアドレスは、1だけインクリメントされる代わりに、単に、それぞれ2又は4だけインクリメントされる。

【0391】本発明において、常に8バイトの倍数が読み取られるが、アドレス発生器は、64バイト未満が読み取られなければならないことを、DRAMインターフェースに送信することができる（これは、ラスタラインの開始または終了に際して必要とされることがある）。これは、スタート及びストップバリューを使用することによって達成される。スタートバリューは、アドレスのトップ部分用に使われる（ビット6以上）そして、読取りが停止されなければならない時を示す信号を生成するために、ストップバリューは、スタートバリューと比較される。

【0392】DRAM信号のエッジを、システムのクロック周期の4分の1の精度に置くために、本発明におけるDRAMインターフェースタイミングブロックはタイミングチェーンを使用する。位相クロックされたループから得られる2つの直交クロックが用いられる。これらのクロックは、組合わされて、概念上の2xクロックを形成する。従って、あらゆる1つのチェーンは、並列接続された2xクロックと反対の位相を持つ2つのシフトレジスタによって作られる。

【0393】まず第一に、ページスタートサイクル用として1つのチェーンがあり、そして、いま1つのチェーンは、リード/ライト/リフレッシュサイクル用である。各サイクルの長さは、マイクロプロセッサインターフェースを介してプログラム可能であり、その後において、ページスタートチェーンは固定した長さを持ち、そして、ページスタートの期間中に、サイクルチェーンの長さは適宜変化する。

【0394】リセットすると、チェーンがクリアされ、そして、パルスが作られる。パルスは、チェーンに沿って移動し、そして、DRAMインターフェースからの状態情報によって方向付けされる。パルスは、DRAMインターフェースクロックを生成する。各DRAMインターフェースクロック周期は、DRAMの1つのサイクルと一致する。従って、DRAMサイクルの長さが異なる場合には、DRAMインターフェースクロックは定速ではない。

【0395】更に、追加タイミングチェーンは、前述のチェーンからのパルスをDRAMインターフェースからの情報と組み合わせ、例えば、notcas、notras、notwe、notbeのような出力ストロブ及びイネイブルを生成する。

12. 予測フィルタ

再び図19、図20、図21において、そして、更に詳細には図40において、時間デコーダの構成図が示され

る。これには、予測フィルタが含まれる。予測フィルタと時間デコーダの残りのエレメントとの間の関係を図26に更に詳細に示す。予測フィルタの構造の本質は、図27及び図37に示される。予測フィルタのオペレーションに関する詳細な記述は、本セクションの「発明の更に詳細な記述」に記載されている。

【0396】一般に、本発明に基づく予測フィルタは、MPEG及びH. 261モードにおいて使用されるが、JPEGモードでは使用されない。JPEGモードにおける場合を思い起こされたい。即ち、時間デコーダは、空間デコーダによって達成される範囲を越えて一切の実質的な復号化を行うことなく、動画フォーマット部を通してデータを供給させるに過ぎない。再び図27を参照することとし、MPEGモードにおいて、フォワード、及びバックワード予測フィルタは同じであり、そして、それぞれのMPEGフォワード、及びバックワード予測ブロックを濾過する。ただし、H. 261はバックワード予測を使わないので、H. 261モードにおいては、フォワード予測フィルタのみが使われる。

【0397】本発明の2つの予測フィルタの各々は、実質的に同じである。再び図17及び505を参照することとし、更に詳細には図37を参照することとする。図には、予測フィルタの構造のブロック図が示される。各予測フィルタは、直列配置された4つのステージを有する。データは、フォーマットステージ505-7に入力され、そして、容易にろ過され得るフォーマットにされる。次のステージ505-2において、I-D予測が、X座標上で行われる。必要な輸送が次元バッファステージ505-3によって行われた後で、I-D予測は、ステージ505-4におけるY座標上で行われる。ステージがどのようにして濾過作用を実施するかについて更に詳細に説明することとする。濾過作用に要求される条件については圧縮規格によって定義されている。H. 261の場合において、現実に行われる濾過作用は、ローパスフィルタの場合に類似する。

【0398】再び図26を参照して、多重規格オペレーションは、MPEG又はH. 261フィルタリングのいずれかを遂行するために予測フィルタは再構成可能であるか、或いは、JPEGモードにおいては全く濾過作用を実施しないことを必要とする。3チップシステムの他の多くの再構成可能な態様と同様に、予測フィルタは、トークンによって再構成される。トークンは、アドレス発生器にオペレーションの特定モードについて通知するためにも使用される。この場合、アドレス発生器は、MPEGとJPEGとの間では著しく変化する必要なデータのアドレスを予測フィルタに供給することができる。

【0399】13. アクセシングレジスタ

マイクロプロセッサインターフェース(MPI)における大抵のレジスタは、当該インターフェースが関連しているステージが停止している場合に限り、修正可能であ

る。従って、レジスタのグループは、一般に、アクセスレジスタと関連している。アクセスレジスタにおける値がゼロである場合には、特定の当該アクセスレジスタと関連しているレジスタのグループは、修正してはならない。1 をアクセスレジスタに記入すると、ステージが停止されることを要求する。ただし、ステージは即座に停止出来ないで、従って、ステージアクセスレジスタは、停止されるまで、値ゼロを保持する。

【0400】MPI と関連し、そして、MPI を介して機能を遂行するために用いられるあらゆるユーザーソフトウェアは、「1 をリクエストアクセスレジスタに記入する後において」アクセスレジスタから1 が読み取られる時まで待たねばならない。ユーザーが1 つの値をコンフィギュレーションレジスタに記入する場合には、これと同時に、そのアクセスレジスタがゼロにセットされるまで、結果は不明である。

【0401】14. マイクロプロセッサインターフェース

空間デコーダ及び時間デコーダにおける全ての回路において、規格バイト幅を持つマイクロプロセッサインターフェース (MPI) が用いられる。MPI は、様々な空間、及び時間デコーダクロックと共に非同期的に動作する。表37を参照して、この表には、このインターフェースにおいて用いられる様々なMPI 信号が示される。信号のキャラクタは、入力出力欄に示され、信号の名前は、信号名欄に示され、そして、信号機能の説明は記述欄に示される。MPI の電気仕様は表38に関して示される。全ての仕様はタイプ別に分類され、そして、記号欄には3つのタイプが示される。これらの記号が何を表すかについては、パラメータ欄に示される。現実の仕様は、それぞれの欄に、最小、最大、及び単位において示される。

【0402】DC 作動条件は、表39に関して示される。欄の見出しは、表38の場合と同じである。DC 電気特性は、表40に関して示され、そして、表38、及び表39の場合と同じ欄題が用いられる。

【0403】15. MPI リードタイミング

MPI リードタイミングダイアグラムのAC特性を、図62に関して示す。図の各ラインは、対応する信号名によってラベル付され、そして、タイミングはナノセカンドで示す。完全なマイクロプロセッサインターフェースのリードタイミング特性は、表41に関して示す。欄番号は、特性欄に記載された信号名に対応する信号を示す。MIN、及びMAXによって識別された欄は、当該信号が利用可能な時間の最大量を示す時間の最小長さを提供する。単位欄は、信号について説明するために用いられた測定単位を示す。

【0404】16. MPI ライトタイミング

MPI ライトタイミングダイアグラムの一般的な記述は、図63に関して示される。この図は、MPI ライト

タイミングと関連した各個別信号名を示す。名前、信号の特性、及び他の様々な物理的特性は、表42に関して示される。

【0405】17. キーホールアドレスロケーション
本発明において、アクセス頻度の小さいメモリーマップは、キーホールレジスタの後ろに配置されている。キーホールレジスタは、それと関連している2つのレジスタを持つ。第1のレジスタは、キーホールアドレスレジスタであり、そして、第2のレジスタは、キーホールデータレジスタである。キーホールアドレスは、拡張アドレススペース内の場所を指定する。キーホールデータレジスタへの読取りまたは書込みオペレーションは、キーホールアドレスレジスタによって指定された場所にアクセスする。キーホールデータレジスタにアクセスした後で、関連キーホールアドレスレジスタはインクリメントする。拡張されたアドレススペース内のランダムアクセスは、各アクセスに対して、キーホールアドレスレジスタに新しい値を書き込むことによってのみ可能である。本発明の範囲内の回路は、複数のキーホールマップを持つこともあり得る。それにもかかわらず、異なるキーホール間の対話はない。

【0406】18. ピクチャエンド

再び図18を参照することとし、この図には、本発明に使用される空間デコーダの全般的な構成図を示す。ピクチャ・エンドの機能に関する記述は、この構成図全体に互って適用される。ピクチャ・エンド機能は、H. 261 符号化ピクチャ情報、MPEG及びJPEG信号を扱うことができるという多重規格における利点を有する。

【0407】既に述べたように、図18の全体的な構成図は、既に説明した2線インターフェースによって相互接続される。各々の機能ブロックは、図17に関して示すステートマシンコンフィギュレーションに基づいて動作するように配列されている。

【0408】一般に、本発明に基づくピクチャ・エンドは、ピクチャ・エンド制御トークンを生成するスタートコード検出器において開始する。ピクチャ・エンド制御トークンは、スタートアップ制御回路を経て、変更されないまま、DRAMインターフェースに供給される。この場合、前記トークンは、DRAMインターフェースにおけるライトスイングバッファをフラッシュするために用いられる。スイングバッファの内容は、このバッファが満杯になった場合に限りRAMに書き込まれることを思い起こされたい。ただし、バッファが満杯でない点においても終了し、ピクチャデータの固着を起こさせることもあり得る。ピクチャ・エンドトークンは、データをスイングバッファの外に強制する。

【0409】本発明は多重規格マシンであるので、このマシンは、各圧縮規格に対して異なつて動作する。更に詳細には、マシン従属アクションサイクルに従って動作する場合について十分に説明することとする。各圧縮規

格に対して、制御トークン、及び／又は、MPUからの出力信号の組合わせにより、利用可能な全アクションサイクルの合計数を選定することが可能であり、或いは、利用可能な全アクションサイクルの合計数は、制御トークン自体の設計によっても選定可能である。この点に関して、本発明は、全ての情報が上流のブロックに収集されてしまうまで、情報が次のブロックに入ることを延期させるように組織されている。データを次のステージまで供給させる準備が整うまで、システムは待機する。このようにして、ピクチャ・エンド信号は、コード化され

たデータバッファに供給され、そして、ピクチャ・エンド信号の制御部分は、データバッファの内容の読み取り引き起こし、そして、ハフマンデコード及び動画デマルチプレクサ回路に供給される。

【0410】ピクチャ・エンド制御トークンの他の利点は、ハフマンデコードデマルチプレクサによる使用に関して、ハフマンデコード及び動画デマルチプレクサ回路に供給される信号の一般的に予測される全範囲、及び／又は、数を持ったことが無い場合であっても、ピクチャの終端をこの制御トークンが識別することである。この条件において、コード化されたデータバッファに保持されている情報は、完全ピクチャとして、ハフマンデコード、及び動画デマルチプレクサに供給される。このようにして、ハフマンデコード、及び動画デマルチプレクサのステートマシンは、システム設計に応じて依然としてデータを扱うことができる。

【0411】ピクチャ・エンド制御トークンの他の利点は、オフチップDRAM、またはスイングバッファに浮遊情報が不用意に残留しないように、コード化データバッファを完全に空にするその能力である。

【0412】ピクチャ・エンド機能の更に他の利点は、エラー回復におけるその用途である。例えば、コード化データバッファに保持されているデータの量が1つの単一画像に関して空間情報を示すために一般的に用いられる量よりも少ないものと仮定すると、最後の画像は、スイングバッファが満杯になるまでデータバッファに保持されるが、しかし、定義によれば、バッファは決して満杯にならない。或る点において、マシンは、エラー条件が退場することを決定する。従って、ピクチャ・エンドトークンは、復号化され、そして、コード化データバッファにおけるデータがハフマンデコード及び動画デマルチプレクサに供給されるように強制する範囲内において、最後の画像は復号化可能であり、そして、情報はバッファから空にされる。結果的に、マシンは、エラー回復モードに入らず、そして、コード化されたデータの処理を首尾よくし継続する。

【0413】ピクチャ・エンドトークンの用途の更に別の利点は、直列パイプラインプロセッサが割り込みなしデータの処理を継続することである。ピクチャ・エンドトークンを使用することにより、直列パイプラインプロ

セッサは、予測された量未満のデータを扱うように構成され、そして、その結果として、処理を継続する。一般に、先行技術によるマシンは、エラー条件に起因して停止する。既に述べたように、コード化データバッファは、その記憶領域にマクロブロックが入来するにつれてこれらをカウントする。更に、ハフマンデコード及び動画デマルチプレクサは、一般に、各画像を復号化する際に予測される情報量を知っている、即ち、ハフマンデコード及び動画デマルチプレクサのステートマシン部分は、各画像回復サイクル期間中に処理するブロック数を知っている。正しい数のブロックがコード化データバッファから到着しない場合には、一般に、エラー回復ルーチンが結果として成立する。ただし、ハフマンデコード及び動画デマルチプレクサを再構成するピクチャ・エンド制御トークンを用いることにより、この制御トークンは、再構成によって、マンデコード及び動画デマルチプレクサに実際に適切な量の情報を扱いつつあることを告げるので、機能し続けることができる。

【0414】再び図17を参照することとし、バッファマネージャのトークンデコード部分は、スタートコード検出器によって生成されたピクチャ・エンド制御トークンを検出する。正常作動状態においては、既に述べたように、スイングバッファの正常なオペレーションに関して、バッファレジスタは、満杯になり、そして、空にされる。再び、データで部分的に満ちているスイングバッファは、スイングバッファが完全に満たされるまで、及び／又は、空になる時間を知る時まで、空にならない。ピクチャ・エンド制御トークンは、バッファマネージャのトークンデコード部分において復号化され、そして、部分的に満ちているスイングバッファをコード化データバッファに空にすることを強制する。これは、究極的に、直接、或いは、DRAMインターフェースを介して、ハフマンデコード、及び動画デマルチプレクサへ供給される。

【0415】19. フラッシングオペレーション

ピクチャ・エンド制御トークンの他の利点は、フラッシュトークンに関連したその機能である。フラッシュトークンは、ステートマシンの再構成制御、或いは、システムへのデータバスのいずれとも関連しない。それどころか、このトークンは、マシン従属のステートマシンによって扱うために前の部分的信号を完成する。各々のステートマシンは、フラッシュ制御トークンを、処理され手はならない情報として認識する。従って、フラッシュトークンは、コード化データバッファの残りの空の部品の全てを満たすために使われ、そして、情報の十分な集合がハフマンデコード及び動画デマルチプレクサに送られることを可能にするために使用される。このように、フラッシュトークンは、バッファに対してパディング様に作用する。

【0416】ハフマン回路におけるトークンデコード

は、フラッシュトークンを認識し、そして、フラッシュトークンがそれに押し込んだ偽りのデータを無視する。次に、ハフマンデコーダは、ピクチャ・エンドトークン及びフラッシュトークンの到着前に存在する場合に最後のピクチャバッファのデータ内容にのみ作用する。ピクチャ・エンドトークン単独の、或いは、フラッシュトークンと組合わされた用途の更なる利点は、ハフマンデコーダ回路の再構成、及び／又は、再組織である。ピクチャ・エンドトークンの到着に際して、ハフマンデコーダ回路は、最後の画像を復号化するために通常予測される情報より少ない情報を持つことを知る。ハフマンデコード回路は、最後の画像に含まれる情報の処理を終了し、そして、この情報をDRAMインターフェースを介して逆モデラーへの出力する。最後の画像の識別に際して、ハフマンデコーダは、クリーンアップモードになり、そして、次の画像情報の到着に対して再調整する。

【0417】20. フラッシュ機能

フラッシュトークンは、本発明に基づき、パイプラインプロセッサ全体を通して供給するため、そして、バッファは空であること、及び他の回路が新しいデータの到着を待つように再構成されるということを保証するために使用される。更に詳細には、本発明は、ピクチャ・エンドトークンの組合せ、パディングワード、及び現行ピクチャのフォームに対する画像処理が完成することを直列パイプラインプロセッサに指示するフラッシュトークンを有する。その後において、様々なステートマシンは、新しく扱うために新しいデータの到着を待つように再構成することを必要とする。更に、フラッシュトークンは、システムに対して、特殊リセットとして作用することに注意されたい。フラッシュトークンは、通過に際して、各ステージをリセットする。しかし、次のステージの処理継続を可能にする。これは、データの損失を防止する。換言すれば、フラッシュトークンは、絶対的リセットとは対照的に可変リセットである。

【0418】21. ストップ・アフタ・ピクチャ

ストップ・アフタ・ピクチャ機能は、そのオペレーションにおける論理的な点において直列パイプラインの伸長回路を開鎖するために用いられる。この時点において、ピクチャ・エンドトークンが生成され、データがデータ入力ラインからの入来を完了、そして、パディングオペレーションが完成したことを示す。パディング機能は、空のデータトークンを部分的に満たす。次に、フラッシュトークンは、生成され、直列パイプラインシステムを通して供給し、そして、レジスタから全ての情報を押し出し、そして、レジスタがそれらの中立スタンバイ状態に戻ることを強制する。次に、ストップ・アフタ・ピクチャイベントが生成され、そして、ユーザ又はシステムのいずれかがこの状態をクリアする時まで、入力は受け入れられない。換言すれば、ピクチャ・エンドトークンはピクチャの終了を送信し、同時に、ストップ・ア

タ・ピクチャオペレーションは、全ての現行処理が終わったことを送信する。

【0419】22. 多重規格サーチモード

本発明の他の特徴は、入来するビットストリームを調べるように直列パイプラインプロセッサへの入力を再構成するために用いられるSEARCHNODE制御トークンの使用である。サーチモードがセットされている場合、スタートコード検出器は、特定のスタートコード、或いは、圧縮規格の任意の1つに使用されるマーカのみを探索する。ただし、他のデータビットストリームからの他のイメージをこの目的のために使用できることを理解されたい。従って、これらのイメージは、本発明全体に互って、制御トークンの組合せ、及び同様の処理を提供するために再構成回路と共にデータトークンを使うことが可能である他の実施例に変更するために使用できる。

【0420】本発明におけるサーチモードの使用は、次に示す場合を含む多くの条件において便利である、即ち、1) データビットストリームの切断が起きた場合、2) ユーザが意図的にチャンネルを変えることによりデータビットストリームを切断する場合、例えば、ケーブル搬送圧縮デジタル動画によるデータの到着、或いは、3) 例えば、光学ディスク、または動画ディスクのような制御可能なデータソースからの高速な順方向または逆方向のユーザ作動化による場合。一般に、サーチモードは、直列パイプラインの正常な処理をマシンが割込みを予測しない点においてユーザが中断する場合に便利である。

【0421】サーチモードのうちのいずれかがセットされている場合、スタートコード検出器は、マシン独立トークンを作るために適した入来するスタートイメージを捜す。規格従属スタートイメージの識別以前にスタートコード検出器に入来する全てのデータは、無意味であり、そして、マシンがこの情報を待っている場合にマシンはアイドル状態であとして、放棄される。

【0422】スタートコード検出器は、数値0コンフィギュレーション構成の任意の1つを仮定可能である。例えば、これらのコンフィギュレーションの1つは、ピクチャのグループ或いはより高位のスタートコードに関する探索を可能にする。このパターンによって、スタートコード検出器は、その全ての入力を放棄し、そして、グループスタート規格イメージを捜す。この種のイメージが識別される場合には、スタートコード検出器は、グループスタートトークンを生成し、そして、サーチモードは自動的にリセットされる。

【0423】1つの単一回路、ハフマンデコーダ、及び動画デマルチプレクス回路は、規格独立セットアップ信号を含む入力信号、並びに、コーディング・スタンダード信号の組合せと共に作動することに注意することが重要である。コーディング・スタンダード信号は、ハフ

マンデコーダ、及び動画デマルチプレクス回路による要請に従って、入来するビットストリームから直接に情報を伝達しつつある。ただし、一方では、ハフマンデコーダ、及び動画デマルチプレクス回路は、規格独立シーケンス信号のオペレーションの下に機能する。

【0424】このオペレーションモードが最も効果的であり、そして、規格従属入力をハフマンデコーダ及び動画デマルチプレクスに運ぶ際に実際の信号自体を運ぶ代りに特殊制御トークンが用いられるように設計できたので、このオペレーションモードが選定された。

【0425】23. 逆モデル

逆モデル化は、これら3つの規格全てのの特徴であり、そして、3つ全ての規格に対して同じである。一般に、トークンバッファにおけるデータトークンは、量子化された係数の値に関する情報、及び表示される係数の間のゼロの数に関する情報を含む（ランレンス符号化の1つの形）。本発明の逆モデルは、トークンと共に使用するように適応させられており、そして、各データトークンが必須条件としての64バリューを含むようにゼロの実行に関する情報を簡単に拡大する。その後における、データトークン内の値は、逆量子化器によって使用できる量子化された係数である。

【0426】24. 逆量子化器

本発明の逆量子化器は、復号化シーケンスにおいて必要とされるエレメントであるが、ICセット全体が多重規格データを扱うことができるような方法において実現された。更に、逆量子化器は、トークンと共に使用できるように適応済みである。逆量子化器は、逆モデルと逆DCT (IDCT) との間に配置される。例えば、本発明において、逆量子化器における加算器は、データがIDOTに移動する以前に、画素デコード数に定数を加えるために使われる。

【0427】IDOTは、情報をコード化するために用いられる各規格に応じて変化する画素復号化数を使う。情報を正しく復号化するために、データがIDCT入りを継続する以前に逆量子化器によって値1024がデコード数に加算される。

【0428】データがIDCTに到達する以前にデータを標準化するために既に逆量子化器内に所在する加算器を使用することにより種々の規格によって圧縮されたデータを扱うためにICに回路またはソフトウェアを追加する必要性を排除する。多重規格オペレーション用の準備整える他のオペレーションは、「量子化後機能」の期間中間に行われ、後で検討することとする。

【0429】データに付随する制御トークンが復号化され、そして、逆量子化器によって行われることが必要な様々な標準化ルーチンは、以下において詳細に識別される。これらの「量子化後」機能は全て、回路の重複を回避し、そして、ICが多重規格復号化データを扱うことができるように実現される。

【0430】25. ハフマンデコーダ、及びパーザ（構文解析器）

再び図18及び図36を参照することとし、空間デコーダは、様々な圧縮規格に従ってハフマンコード化されたデータを復号化するためのハフマンデコーダを有する。

規格JPBG、MPEG、及びH.261の各々は、特定のデータがハフマン符号化されていることを要求するが、一方において、各規格が要求するハフマン復号化法の相違は重要な意味をもつ。本発明の空間デコーダにお

10 いては、各規格に対して1つずつ合計3つの個別ハフマンデコーダを設計して作成する事なく、本発明は、各ハフマンデコーダの共通態様を識別し、そして、これらの共通態様のみを1度だけ作成することにより、貴重なダイスペースを節約する。更に、賢明な多重部品アルゴリズム、即ち、他の規格に対して共通性を持つ各ハフマンデコーダの態様の共通性を一層強化するようなアルゴリズムが用いられる。要約すれば、ハフマンデコーダ321は、図36に示す他のユニットと共に作動する。これらの他のユニットとは、パーザステートマシン322、インシフタ323、データインデックスユニット324、ALU325、及びトークンフォーマッティング部326である。既に述べたように、これらのブロックの間の接続は、2線インターフェースによって管理される。これらのユニットがどのように機能するかということについては続いて更に詳細に記述することとし、ここでは、本発明に基づくハフマンデコーダの多重規格オペレーションを支援する特定の態様に焦点を合わせて説明する。

30 【0431】本発明のパーザステートマシンは、動画パーザの他のブロックのオペレーションを統合するように作用するプログラム可能なステートマシンである。パーザステートマシンは、データに回答して、データと並列配置されて他のブロックへ供給される制御ワードを生成することにより、この制御ワードが作用する他のブロックを制御する。ブロックは2線インターフェースを介して接続されるので関連データと並置して制御ワードを供給することは、有用であるばかりでなく、必須である。このように、データとコントロールは同時に到着する。制御ワードの通過は、制御ラインによって図36に示すように、ブロックを接続するデータラインの下を通る。特にこのコードワードは、復号化されつつある特定の規格を識別する。

【0432】ハフマンデコーダは、ある種の制御機能も遂行する。特に、ハフマンデコーダ321は、データインデックス部324及びALU325の特定の機能を制御できるステートマシンを含む。ハフマンデコーダによるこれらのユニットの制御は、ブロックレベル情報を正しく復号化するために必要である。パーザステートマシン322にこれらの決定を行わせると時間がかかり過ぎる。

【0433】本発明のハフマンデコーダの重要な態様は、データビットをハフマンデコーダに読み取る際にコード化されたデータビットを反転する能力である。この能力は、H. 261ハフマンスタイルコードを復号化するために必要とされる。理由は、H. 261（実質的には、MPEG）によって用いられる特定タイプのハフマンコードは、JPEGによって用いられるコードと反対の極性を持つことによる。従って、インバータを使用することにより、ハフマンデコーダが使用する表と同一の表を3つ全ての規格に対して実質的に使用可能にする。ハフマンデコーダが3つ全ての規格を実現する方法の他の態様については、「発明の更に詳細な説明」のセクションにおいて更に詳細に説明する。

【0434】データインデックスユニット324は、多重部品アルゴリズムの第2の部分を実行する。このユニットは、実際のハフマン復号化データを提供するルックアップテーブル（参照用表）を含む。この表へのエントリは、ハフマンデコーダによって生成されたインデクス数に基づいて組織される。

【0435】ALU325は、多重部品アルゴリズムの残りの部分を実現する。特に、ALUは、符号拡張を扱う。更に、ALUは、ベクトル予測及びDC予測を保持するレジスタファイルを含む。このファイルに使用については、予測フィルタに関係したセクションにおいて記述する。更に、ALUは、空間デコーダによって復号化されつつあるピクチャの構造を介してカウントするカウンタを含む。特に、ピクチャの次元は、カウンタと関連するレジスタにプログラムされ、「ピクチャ・スタート」、及びマクロ・ブロックコード開始の検出を容易にする。

【0436】本発明に基づき、トークンマッチング部326（TF）は復号化されたデータをデータトークンにアセンブルする。これらのトークンは、次に、時空間デコーダにおける残りのステージまたはブロックに供給される。

【0437】本発明において、インシフタはスタートコード検出器を供給するデータを緩衝するFIFOからデータを受け取る。インシフタが受け取るデータには、一般に、2つのタイプがある、即ち、「トークン」のセクションにおいて更に検討されるように、データトークン、及びスタートコード検出器によってそれぞれのトークンと交換されるスタートコードである。データの殆どは、復号化を必要とするデータトークンであることに注意されたい。

【0438】インシフタは、ハフマンデコーダに直列的にデータを供給する一方、制御トークンを並列に供給する。ハフマンデコーダにおいて、ハフマン符号化されたデータは、多重部品アルゴリズムの第1の部分に従って復号化される。特に、特定のハフマンコードは、識別され、次に、インデクス数と交換される。

【0439】更に、ハフマンデコーダは、図36に示す他のブロックによる特別な扱いを必要とする特定のデータを識別する。このデータは、ブロックエンド及びエスケープを含む。本発明において、時間は、データインデックスユニットにおいてではなく、ハフマンデコーダにおいてこれらを検出することによって節約される。次に、このインデクス数は、データインデックスユニット324に供給される。データインデックスユニットは、本質的には、ルックアップテーブルである。アルゴリズムの1つの態様に基づくルックアップテーブルは、JPEGによって指定されたハフマンコードテーブルと大差無い。一般に、概して、それは、JPEGが代替JPEG表を転送するために指定するフォーマットは圧縮データフォーマットである。

【0440】復号化されたインデクス数、または他のデータは、データインデックスユニット324から、付随制御ワードと共に、既に述べたオペレーションを実施するALU 504-5に供給される。

【0441】データ及び制御ワードは、ALU325からトークンフォーマット部326（TF）に供給される。トークンフォーマット部において、データは、必要に応じて、トークンを形成するために、制御ワードと組み合わせられる。次に、トークンは、空間デコーダのその次のステージに運ばれる。この時点において、システムによって使われると同数のトークンがあることに注意されたい。

【0442】2.6. 離散逆コサイン変換

本発明に基づく離散逆コサイン変換（IDCT）は、当該ピクチャのDC成分の周波数に関係したデータを伸長する。特定のピクチャが圧縮されつつある場合、ピクチャにおける光の周波数（振動数）は量子化され、記憶する必要がある情報の全体量を減少させる。IDCTは、この量子化されたデータを用いて、それを伸長し、周波数情報に戻す。

【0443】IDCTは、サイズが8×8ピクセルであるピクチャの部分に作用する。このデータに適用された数学は、データをコード化するために用いられる特定の規格によって主として管理される。ただし、本発明における重要な用途は、回路の不必要な重複を回避するために、規格の間に共通の数学的機能を作ることである。

特定のスケールリングオーダを用いることにより、アルゴリズムの上部と下部との間の対称性を増大させ、それによって、回路の追加必要性を排除する共通数学機能が再使用できる。

【0444】IDCTは、多数の多重規格トークンに回答する。IDCTの第1の部分は、処理に対してデータトークンのサイズが正しいことを保証するために、入来データをチェックする。実際、エラーがさほど大きくないならば、トークンストリームは、条件により訂正可能である。

【0445】27. バッファマネジャ

本発明のバッファマネジャは、入来動画情報を受け取り、そして、データ到着のタイミング、ディスプレイ、及びフレームレートに関する情報をアドレス発生器に供給する。多重バッファは、表示およびディスプレイの両レートの変更を可能にするために用いられる。表示およびディスプレイレートは、一般に、コード化されたデータ、及び情報がディスプレイされつつあるモニタに従って変化する。データ到着レートは、一般に、符号化、復号化、またはデータを作るために用いられる原材料に含まれる誤差に応じて変化する。情報は、バッファマネジャに到着すると、伸長される。ただし、データは、伸長回路にとって有用であるが、使用中の特定のディスプレイユニットにとっては有用でないような順序に配列される。データのブロックがバッファマネジャに入力されると、バッファマネジャは、ディスプレイ装置が使用できる順序にデータのブロックが配置できるように、情報をアドレス発生器へ供給する。これにより、バッファマネジャは、使用中の特定ディスプレイ装置にデータブロックを表示できるように入来するデータブロックを調節するために必要なフレームレート変換を用いる。

【0446】本発明において、バッファマネジャは、情報を主としてアドレス発生器に供給する。ただし、システムの他のエレメントをインターフェースすることも必要である。例えば、結果としてこれらのトークンを書込みアドレス発生器に供給するバッファマネジャへトークンを転送する入力FIFOとのインターフェースがある。

【0447】更に、バッファマネジャは、ディスプレイアドレス発生器ともインターフェイスし、ディスプレイ装置が新しいデータをディスプレイする準備が整っているかどうかに関する情報を受け取る。同様に、バッファマネジャは、バッファからのディスプレイ用の情報をディスプレイアドレス発生器がクリアしたことを確認する。

【0448】本発明のバッファマネジャは、特定のバッファが空であるか、満ちているか、使用準備が整っているか、或いは、使用中であるかどうかを記憶し、更に、各バッファにおける特定データと関連する表示数を記憶する。このようにして、バッファマネジャは、一時にただ1つのバッファをディスプレイ準備が整った状態にすることによって、部分的に、バッファの状態を決定する。いったん、バッファがディスプレイされると、バッファは、「空の」状態にある。バッファマネジャは、ピクチャ・スタート、フラッシュ、有効なトークン、またはアクセストークンを受け取ると、各バッファの状態、及び新しいデータの受け入れ準備について決定する。例えば、ピクチャ・スタートトークンは、新しいデータを受け入れることが可能なバッファを見付けるために、バッファマネジャに各バッファを経て循環させる。

【0449】更に、バッファマネジャは、それが受け取るトークンによって指示される多重規格必要条件を扱うように構成可能である。例えば、H. 261規格標準において、ディスプレイ期間中に、データは、スキップ可能である。この種のトークンがバッファマネジャに到着した場合には、スキップされるべきデータは、それが記憶されているバッファからフラッシュされる。

【0450】このように、バッファを管理することによって、データは、当該データをコード化するために使われる圧縮規格、データが復号化される速度、及び使用中の特定タイプのディスプレイ装置に従って効果的にディスプレイされることが可能である。

【0451】今までの記述は、当該技術分野における通常の熟練者にとって、全ての付随的な特徴、目的、及び利点を備えた本発明を実現させるに充分な程度の詳細さを以て、全体的な概念、システムの実現、及び本発明の様々な態様のオペレーションについて適切に説明していると確信する。ただし、本発明、及び本発明の様々な実施例の一層具体的かつ商業的な実現方法と関連した追加的な詳細を更に徹底的に理解することを容易にするためには、一層の記述及び説明を以下に継続して示すことが好ましい。

【0452】以下は多重規格動画デコーダのチップ・セットのより詳細な説明である。説明はA、B、Cの3つの部分に分かれる。

【0453】説明の組織化、明瞭化および便利化のために、追加開示を以下のセクションに再度示すこととする。

【0454】・チップ・セットのチップに共通の特徴に関する記述。

【0455】・トークン

・2線インターフェース

・DRAMインターフェース

・マイクロプロセッサインターフェース

・クロック

・空間デコーダチップの記述

・時間デコーダチップの記述

セクション A. 1 「本記述の利用方法」

第1の記述セクションは、チップ・セットの使用に関連した電気設計上の問題の大部分を対象とする。

【0456】A. 1. 1 「印刷上の慣例」

或る種のクラスの情報を強調するために多少の印刷上の慣例が用いられる。その例を次に示す。

【0457】トークンの名称

ワイヤネーム・アクティブハイ信号

ワイヤネーム・アクティブロー信号

レジスタネーム

セクション A. 2 「動画デコーダファミリー」

* 30MHz オペレーション

* デコードMPEG、JPEG & H. 261

- * 25 Mb/s に対するコード化されたデータレート
- * 21 MB/s に対する動画データレート
- * MPEG 分解能 704×480、30 Hz、4:2:0 まで
- * 融通性を持つクロマサンプリングフォーマット
- * 全体 JPEG 基底線復号化
- * グルーレスページモード DRAMA インターフェース
- * 208 のピン PQFP パッケージ
- * 独立コード化データ、及びデコードクロック
- * 再オーダ MPEG ピクチャシーケンス

動画デコードファミリは、高分解能度デジタル動画デコードを実現するための低チップカウント解決方法を提供する。現在、チップ・セットは、3つの異なる動画及びピクチャ符号化システム、即ち、JPEG、MPEG、及び H. 261 をサポートするように構成可能である。

【0458】全 JPEG 基底線ピクチャ復号化がサポートされる。720×480、30 Hz、4:2:2 JPEG 符号化動画がリアルタイムに復号化可能である。

【0459】CIF (共通交換フォーマット)、及び QCIF H. 261 動画が復号化可能である。740×480、30 Hz、4:2:0 までのフォーマットを持つ全特機能 MPEG 動画が復号化可能である。

【0460】注記：前記の値は、実例として示したに過ぎず、そして、必ずしも本発明の 1 実施例の制限条件を示すものではない。従って、他の値、及び/又は、範囲を使用しても差し支えないことを理解されたい。

【0461】A. 2. 1 「システムコンフィギュレーション」

A. 2. 1. 1 「出力フォーマット化」

以下に示す各々の例において、出力フォーマット部は、場合によっては、空間デコードまたは時間デコードの出力に供給されるデータを用いること、そして、コンピュータ又はディスプレイシステムに対してフォーマットし直すことが要求される。このフォーマット化の詳細は、アプリケーションにより異なる。一例として簡単な場合には、デコードチップによってブルックフォーマット化されたデータ出力を用いて、そして、そのデータ出力をラスタ順に書込むためのアドレス発生器を必要とするだけである。

【0462】イメージフォーマット部は、広い範囲に亘って出力フォーマット化機能を提供する 1 つの単一チップ VLSI デバイスである。

【0463】A. 2. 1. 2 「JPEG 静止画復号化」

オフチップでない DRAM を備えた 1 つの単一空間デコードは、基底線 JPEG イメージを高速に復号化可能である。空間デコードは、基底線 JPEG の全ての機能をサポートする。ただし、復号化可能なイメージサイズは、ユーザーによって供給された出力バッファ

のサイズによって制限されこともあり得る。出力フォーマット部の特性は、クロマサンプリングフォーマット、及びサポート可能な色空間を制限することもあり得る。

【0464】A. 2. 1. 3 「JPEG 動画復号化」
空間デコードにオフチップ DRAM を加えることにより、JPEG 符号化動画をリアルタイムに復号化することが可能である。必要とされるバッファのサイズ及びスピードは、動画及びコード化データのレートに依存する。時間デコードは、JPEG 符号化動画を復号化する必要はない。ただし、時間デコードが多重規格デコードチップ・セット上に在る場合に、システムが JPEG オペレーション用に構成されている場合には、変更または修正なしで、時間デコードを介してデータを単に供給するに過ぎない。

【0465】A. 2. 1. 4 「H. 261 復号化」
空間デコード及び時間デコードは、H. 261 動画デコードを実現するために両方共必要である。双方のデバイス上の DRAM インターフェースは、小さいピクチャフォーマットにおいて低コード化データレートで作動する場合に、適切なオペレーションのために必要とされる DRAM の量を軽減することができるよう構成可能である。一般に、各々の空間デコード及び時間デコードは、1 つの単一 4 Mb (例えば、512 k×8) DRAM を必要とする。

【0466】A. 2. 1. 5 「MPEG 復号化」
MPEG オペレーションのために必要とされるコンフィギュレーションは、H. 261 の場合と同じである。ただし、当該技術分野における通常の熟練者であれば理解できるように、MPEG において利用可能な更に大きいピクチャフォーマットをサポートするためには、更に大きい DRAM バッファが要求されることもあり得る。

【0467】セクション A. 3 「トークン」

A. 3. 1 「トークンフォーマット」

本発明に従う場合、トークンは、デコードチップ・セットを介して情報を伝達するために、拡張可能なフォーマットを提供する。本発明においては、トークンの各ワードの最小幅は 8 ビットであるが、当該技術分野における通常の熟練者であれば、トークンの幅は任意であっても差し支えないことを理解するはずである。更に、1 つのトークンは、1 又は複数のワードに亘って拡張可能であり、これは、各ワードにおける拡張ビットを用いて達成される。

【0468】拡張ビットは、トークンが他のワード内に継続するかどうかを示す。トークンの最後のワード以外の全てのワードは 1 にセットされる。トークンの第 1 のワードが 0 の拡張ビットを持つ場合には、これは、当該トークンの長さがわずかに 1 ワードであることを示す。

【0469】各トークンは、それがトークンの第 1 ワードのビット 7 においてスタートするアドレスフィールド

10

20

30

40

50

によって識別される。アドレスフィールドの長さは可変であり、そして、複数のワードに亘って潜在的に拡張可能である（ただし、現在のチップにおいては長さが8ビットを超過するアドレスは無いので、当該技術分野における通常の熟練者であれば、この場合にも、アドレスの長さは任意であり得ることを理解するはずである）。

【0470】インターフェースによっては、データの8よりも多いビットを転送する。例えば、空間デコーダの出力の幅は9ビットである（拡張ビットを含み10ビット）。これらの追加ビットを利用する唯一のトークンはデータトークンである。データトークンは、システムの特定場所において処理を実施するために必要なビットと同数のビットを持つことができる。他の全てのトークンは、追加ビットを無視する。

【0471】A. 3. 2 「データトークン」

データトークンは、1つの処理ステージから次のステージにデータを運ぶ。従って、このトークンの特性は、デコーダを通過するにつれて変化する。更に、データトークンによって運ばれるデータの意味は、当該データトークンがシステム内のどこにあるかによって変化する、すなわち、データは位置依存性である。この点に関して、データは、当該データトークンが空間デコーダ内のどこに所在するかに応じて周波数領域データ又は画素領域データのいずれかであり得る。例えば、空間デコーダの入力において、データトークンは、8ビットのワードにパックされたビット直列コード化動画データを持つ。この点においては、各トークンの長さには制限がない。ただし、対照的に、空間デコーダの出力においては、各データトークンは、各ワードの幅が9ビットであるちょうど64のワードを持つ。

【0472】A. 3. 3 「トークンフォーマット化デ

ータの使用」

用途によっては、回路は、デコーダ又はチップ・セットの入力または出力に直接接続することが必要である。大抵の場合、データトークンを収集し、そして、同期化情報（例えば、PICTURE STARTのような）を提供する2、3のトークンを検出することで充分である。この点に関しては、以下のセクションA. 16「空間デコーダの出力への接続」及びA. 19「時間デコーダの出力への接続」を参照されたい。

10 【0473】既に述べたように、各々の新しいトークンが何時スタートするかを識別するためには、拡張ビットに関するアクティビティを観察するだけで充分である。再び、拡張ビットは、現行トークンの最後のワードを送信する。更に、トークンを識別するために、アドレスフィールドをテストすることが出来る。不必要または認識されなかったトークンは、それらの内容を知ることなしに、消費可能（そして、破棄可能）である。ただし、認識されたトークンは、適切なアクションを起こさせる。

20 【0474】更に、空間デコーダへのデータ入力は、コード化されたデータのバイトとして、或いは、データトークンにおいて供給可能である（セクションA. 10「コード化データ入力」参照）。コード化データポート、或いは、マイクロプロセッサインターフェースを介してトークンを供給することは、デコーダチップ・セットの多数の特徴がデータストリームから構成されることを可能にする。これは、マイクロプロセッサインターフェースを介してコンフィギュレーションを実施することの代替案を提供する。

【0475】

30 【表1】

7	6	5	4	3	2	1	0	トークン名称	備考
0	0	1						QUANT SCALE	
0	1	0						PREDICTION MODE	
0	1	1						(予約済)	
1	0	0						MVD FORWARDS	
1	0	1						MVD BACKWARDS	
0	0	0	0	1				QUANT TABLE	
0	0	0	0	0	1			DATA	
1	1	0	0	0	0			COMPONENT NAME	
1	1	0	0	0	1			DEFINE SAMPLING	
1	1	0	0	1	0			JPEG TABLE SELECT	
1	1	0	0	1	1			MPEG TABLE SELECT	
1	1	0	1	0	0			TEMPORAL REFERENCE	
1	1	0	1	0	1			MPEG DCH TABLE	
1	1	0	1	1	0			(予約済)	
1	1	0	1	1	1			(予約済)	
1	1	1	0	0	0	0		(予約済) SAVE STATE	
1	1	1	0	0	0	1		(予約済) RESTORE STATE	
1	1	1	0	0	1	0		TIME CODE	
1	1	1	0	0	1	1		(予約済)	
0	0	0	0	0	0	0	0	NULL	
0	0	0	0	0	0	0	1	(予約済)	
0	0	0	0	0	0	1	0	(予約済)	
0	0	0	0	0	0	1	1	(予約済)	
0	0	0	1	0	0	0	0	SEQUENCE START	
0	0	0	1	0	0	0	1	GROUP START	
0	0	0	1	0	0	1	0	PICTURE START	
0	0	0	1	0	0	1	1	SLICE START	
0	0	0	1	0	1	0	0	SEQUENCE END	
0	0	0	1	0	1	0	1	CODING STANDARD	
0	0	0	1	0	1	1	0	PICTURE END	
0	0	0	1	0	1	1	1	FLUSH	
0	0	0	1	1	0	0	0	FIELD INFO	

表A.31 トークンの概要(1/2)

【0476】

【表2】

7	6	5	4	3	2	1	0	トークン名称	備考
0	0	0	1	1	0	0	1	MAX COMP ID	
0	0	0	1	1	0	1	0	EXTENSION DATA	--
0	0	0	1	1	0	1	1	USER DATA	
0	0	0	1	1	1	0	0	DHT MARKER	
0	0	0	1	1	1	0	1	DQT MARKER	
0	0	0	1	1	1	1	0	(予約) DNL MARKER	
0	0	0	1	1	1	1	1	(予約) DRI MARKER	
1	1	1	0	1	0	0	0	(予約)	
1	1	1	0	1	0	0	1	(予約)	
1	1	1	0	1	0	1	0	(予約)	
1	1	1	0	1	0	1	1	(予約)	
1	1	1	0	1	1	0	0	BIT RATE	
1	1	1	0	1	1	0	1	VBV BUFFER SIZE	
1	1	1	0	1	1	1	0	VBV DELAY	
1	1	1	0	1	1	1	1	PICTURE TYPE	
1	1	1	1	0	0	0	0	PICTURE RATE	
1	1	1	1	0	0	0	1	PEL ASPECT	
1	1	1	1	0	0	1	0	HORIZONTAL SIZE	
1	1	1	1	0	0	1	1	VERTICAL SIZE	
1	1	1	1	0	1	0	0	BROKEN CLOSED	
1	1	1	1	0	1	0	1	CONSTRAINED	
1	1	1	1	0	1	1	0	(予約) SPECTRAL LIMIT	
1	1	1	1	0	1	1	1	DEFINE MAX SAMPLING	
1	1	1	1	1	0	0	0	(予約)	
1	1	1	1	1	0	0	1	(予約)	
1	1	1	1	1	0	1	0	(予約)	
1	1	1	1	1	0	1	1	(予約)	
1	1	1	1	1	1	0	0	HORIZONTAL MBS	
1	1	1	1	1	1	0	1	VERTICAL MBS	
1	1	1	1	1	1	1	0	(予約)	
1	1	1	1	1	1	1	1	(予約)	

表A.3.1 トークンの概要 (2/2)

A. 3. 4 「トークンの説明」

このセクションにおいては、本発明に基づき、空間デコーダ及び時間デコーダチップにおいて実現されるトークンについて詳しく説明する。

注記：「r」は、現在予約されているビットを意味し、

そして、値は0である。特記されない限り、全ての整数は符号無しである。

【0477】

【表3】

記 述									
1	1	1	1	0	1	1	0	0	
1	r	r	r	r	r	r	b	b	
1	b	b	b	b	b	b	b	b	
0	b	b	b	b	b	b	b	b	
BIT RATE テストのみ									
MPEGビットレートパラメータRをもつ。MPEGビットストリーム解読時に生成される。									
b: MPEGにより定義された18ビット整数									
1	1	1	1	1	0	1	0	0	
0	r	r	r	r	r	r	c	b	
BROKEN CLOSED									
2つのMPEGフラグを持つ。									
c: クローズド_ギャップ									
b: ブロークン_リンク									
1	0	0	0	1	0	1	0	1	
0	s	s	s	s	s	s	s	s	
CODING STANDARD									
s: 現行符号化を示す8ビット整数。現在該当されている値を次に示す。									
0: H.261									
1: JPEG									
2: MPEG									
1	1	1	0	0	0	0	c	c	
0	n	n	n	n	n	n	n	n	
COMPONENT NAME									
コンポネントIDとコンポネント名との間の関係を連絡する。更に以下、参照・・・									
c: 2ビットコンポネントID									
n: 8ビットコンポネント名									
1	1	1	1	1	0	1	0	1	
0	r	r	r	r	r	r	r	c	
CONSTRAINED									
c: MPEGビットストリームから解読された制約パラメータフラグを持つ。									

表A.3.2 空間デコーダ及び時間デコーダにおいて実現されるトークン(1/8)

【0478】

【表4】

7	6	5	4	3	2	1	0	記 述
1	0	0	0	0	0	1	c c	DATA
1	d	d	d	d	d	d	d	デコーダチップセットを介してデータを送る。
								c: 2ビット整数コンポネントID (A.3.5.1参照)。
0	d	d	d	d	d	d	d	フィールドはコード化データを持つトークンに関して定義されていない(ピクセル情報以外)
1	1	1	1	1	0	1	1	DEFINE MAX SAMPLING
1	r	r	r	r	r	r	h h	最大水平及び垂直サンプリング。これらはマクロブロックのあらゆるコンポネントの水平・垂直ブロックの最大数を示す。A.3.5.2参照
0	r	r	r	r	r	r	v v	h: 2ビット水平サンプリング番号 v: 2ビット垂直サンプリング番号
1	1	1	0	0	0	1	c c	DEFINE SAMPLING
1	r	r	r	r	r	r	h h	特定色成分に対する水平及び垂直サンプリング番号。
0	r	r	r	r	r	r	v v	A.3.5.2参照。 c: 2ビット成分ID h: 2ビット水平成分番号 v: 2ビット垂直成分番号
0	0	0	0	1	1	1	0	DHT MARKER
								このトークンは、ビデオデマックスに、後続するDATAトークンが、JPEGが定義したハフマン表セグメント構文を用いて作成されたハフマン表の仕様を含むことを通知する。このトークンはコード化規格がJPEGとして構成されている場合に限り有効である。このトークンはデータストリーム中において、DHTマーカに遭遇した場合、JPEG解読期間中にスタートコード検出器によって生成される。

表A.3.2 空間デコーダ及び時間デコーダにおいて実現されるトークン(2/3)

[0479]

[表5]

121

122

E	7	6	5	4	3	2	1	0	記 述
0	0	0	0	1	1	1	1	0	DNL MARKER このトークンは、ビデオデマックスに、後続するトークンがフレーム内の行数を規定する J P E G パラメータ N L を含むことを通知する。このトークンはデータストリーム中において D N L マーカに遭遇した場合、J P E G 解読期間中にスタートコード検出器によって生成される。
0	0	0	0	1	1	1	0	1	DQT MARKER このトークンは、ビデオデマックスに、後続する D A T A トークンが、J P E G が定義した量子化表セグメント構文を用いて作成された仕様を含むことを通知する。このトークンはコード化規格が J P E G として構成されている場合に限り有効である。ビデオデマックスは新しい量子化表情報を含む Q U A N T _ T A B L E トークンを生成する。このトークンはデータストリーム中において D Q T に遭遇した場合、J P E G 解読期間中にスタートコード検出器によって生成される。
0	0	0	0	1	1	1	1	1	DRI MARKER このトークンは、ビデオデマックスに、後続するトークンが、再開始マーカ間の最小符号化ユニット数を規定する J P E G パラメータ R i を含むことを通知する。このトークンは、データストリーム中において D R I マーカに遭遇した場合、J P E G 解読期間中にスタートコード検出器によって生成される。

表 A. 3.2 空間デコード及び時間デコードにおいて実現されるトークン (3/9)

【 0 4 8 0 】

30 【表 6】

E	7	6	5	4	3	2	1	0	記 述
1	0	0	0	1	1	0	1	0	EXTENSION_DATA_JPEG
0	v	v	v	v	v	v	v	v	このトークンは、ビデオデマックスに、後続するトークンが拡張データを含むことを通知する。A.11.3 [スタートコードからトークンへの変換] 及びA.14.6 [ユーザ及び拡張データの受取り] 参照。 JPEG作動期間中、8ビットフィールドはJPEG マーカ値を持つ。これにより、拡張データのクラスが 識別可能となる。
0	0	0	0	1	1	0	1	0	EXTENSION_DATA_MPEG
									このトークンは、ビデオデマックスに後続するDATA トークンが、拡張データを含むことを通知する。 A.11.3 [スタートコードからトークンへの変換] 及びA.14.6 [ユーザ及び拡張データの受取り] 参照
1	0	0	0	1	1	0	0	0	FIELD_INFO
0	r	r	r	t	p	i	f	f	そのディスプレイを支援するために後続するピクチャ に関する情報を持つ。この機能は一切の既存コード化 規格により合図されない。 t: ピクチャがインターレースされたフレームの場合には このビットはアップフィールドが第1(t=0) 又は第2 であるかを示す。 p: ピクチャがフィールドの場合は次のピクチャが当該 フレーム上側であるか(p=0) 又は下側であるかを示す f: 3ビット数であり8フィールドPALシーケンスに おけるフィールド位置を示す。
0	0	0	0	1	0	1	1	1	FLUSH
									現行コード化データの終端を示しデコードを介し、デ ータストリームの終端を押すために用いられる。
0	0	0	0	1	0	0	0	1	GROUP_START
									解読MPEG時にピクチャのグループスタートコード が見付かった場合、又は解読JPEG時にフレームマ ーカが発見された場合に生成される。

表A.3.2 空間デコード及び時間デコードにおいて実現されるトークン(4/9)

[0481]

[表7]

E	7	6	5	4	3	2	1	0	記 述
1	1	1	1	1	1	1	0	0	HORIZONTAL_MBS
1	r	r	r	h	h	h	h	h	h; マクロブロックにおけるピクチャの水平幅を示すビット数13の整数。
0	h	h	h	h	h	h	h	h	
1	1	1	1	1	0	0	1	0	HORIZONTAL_SIZE
1	h	h	h	h	h	h	h	h	h; 画素におけるピクチャの水平幅を示すビット数16の整数。これは任意の整数でも受け支えない。
0	h	h	h	h	h	h	h	h	
1	1	1	0	0	1	0	c	c	JPEG_TABLE_SELECT
0	r	r	r	r	r	r	t	t	規定された色成分に対してどの量子化表を使用するかを逆量子化器に報告する。 c: 2ビット成分 (A.3.5.1参照) t: 2ビット整数表番号
1	0	0	0	1	1	0	0	1	MAX_COMP_ID
0	r	r	r	r	r	r	r	r	m: 次のピクチャに使用されるべき成分IDの最大値 (A.3.5.1参照) を示す2ビット整数。
0	1	1	0	1	0	1	c	c	MPEG_DCH_TABLE
0	r	r	r	r	r	r	t	t	色成分cc用どのDC係数ハフマン表を使用するべきかを構成する。 c: 2ビット成分 (A.3.5.1参照) t: 2ビット整数表番号
0	1	1	0	0	1	1	d	n	MPEG_TABLE_SELECT
									イントラ又は非イントラ情報用にデフォルト又はユーザ定義量子化表を使用するかどうかを逆量子化器に通知する。 n: 0はイントラ情報を示す。1は非イントラ d: 0はデフォルト表を示す。1はユーザ定義

表A.3.2 空間デコード及び時間デコードにおいて実現されるトークン(5/9)

【0482】

【表8】

E	7	6	5	4	3	2	1	0	記 述
1	1	0	1	d	v	v	v	v	MVD_BACKWARDS
0	v	v	v	v	v	v	v	v	後方モーションベクトルの1つの成分 (垂直又は水平のどちらか) を持つ。 d: 0 はx成分を示す。1 はy成分 v: 12ビットの2の補数。LSBは半画素解像度を 提供する。
1	1	0	0	d	v	v	v	v	MVD_FORWARDS
0	v	v	v	v	v	v	v	v	順方向モーションベクトルの1つの成分 (垂直又は水平のどちらか) を持つ。 d: 0 はx成分を示す。1 はy成分 v: 12ビットの2の補数。LSBは半画素解像度を 提供する。
0	0	0	0	0	0	0	0	0	NULL 何もしない
1	1	1	1	1	0	0	0	1	PEL_ASPECT
0	r	r	r	r	p	p	p	p	p: MPEGより定義された4ビット整数
0	0	0	0	1	0	1	1	0	PICTURE_END
									実行ピクチャの終端を示すためにスタートコード検出器により挿入。
1	1	1	1	1	0	0	0	0	PICTURE_RATE
0	r	r	r	r	p	p	p	p	p: MPEGにより定義された4ビット整数。
1	0	0	0	1	0	0	1	0	PICTURE_START
0	r	r	r	r	n	n	n	n	新しいピクチャのスタートを示す。 n: スタートコード検出器によってピクチャに割当てられた4ビット画像インデックス

表A.3.2 空間デコーダ及び時間デコーダにおいて実装されるトークン(6/9)

【0483】

【表9】

E	7	6	5	4	3	2	1	0	記 述
1	1	1	1	0	1	1	1	1	PICTURE_TYPE MPEG
0	r	r	r	r	r	r	p	p	p: 後続するピクチャのピクチャコード化タイプを示す 2ビット整数。 0: イントラ 1: 予測 2: 双方向性予測 3: DCイントラ
1	1	1	1	0	1	1	1	1	PICTURE_TYPE H.261
1	r	r	r	r	r	r	0	1	種々のH.261オプションがオン(1)又はオフ(0)であることを示す。これらのオプションはMPEG及びJ PEGに対して常にオフである。 s: スプリットスクリーンインジケータ d: ドキュメントカメラ f: フリーズピクチャリリース ソースピクチャフォーマット q=0: CCIF q=1: CIF
0	0	1	0	h	y	x	b	f	PREDICTION_MODE
									後続するマクロブロックに対する予測モードを示すフラグビットの集合。 f: 順方向予測 b: 後方向予測 x: リセット順方向ベクトルインジケータ y: リセット後方向ベクトルインジケータ h: H.261フィルタ使用可能化
0	0	0	1	s	s	s	s	s	QUANT_SCALE
									逆量子化器に新しいスケール係数を通知する。 s: レンジ 1...31 における5ビット整数。 値 0は予約済み

表A.3.2 空間デコーダ及び時間デコーダにおいて実現されるトークン(7/9)

【0484】

【表10】

E	7	6	5	4	3	2	1	0	記 述
1	0	0	0	0	1	r	t	t	QUANT_TABLE
1	q	q	q	q	q	q	q	q	規定された逆量子化表に54の8ビット符号無し整数をロードする。値はジグザグ順である。
0	q	q	q	q	q	q	q	q	t; ロードされるべき逆量子化表に指定する2ビット整数。
0	0	0	0	1	0	1	0	0	SEQUENCE_END
									MPEGシーケンス_エンド_コード及びJPEG EOIマーカはこのトークンを生成させる。
0	0	0	0	1	0	0	0	0	SEQUENCE_START
									MPEGシーケンス_スタート_コードにより生成される。
1	0	0	0	1	0	0	1	1	SLICE_START
0	s	s	s	s	s	s	s	s	MPEGスライス_スタート、H.261 GOB及びJPEG再同期化インターバルに対応する。8ビット整数
									's'の解釈はコード化規格によって異なる。
									MPEG; スライス垂直位置+1
									H.261; ブロックグループ番号+1
									JPEG; 再同期化インターバル識別 (4つのLSBのみ)
1	1	1	0	1	0	0	1	1	TEMPORAL_REFERENCE
0	t	t	t	t	t	t	t	t	t; 時間リファレンスを持つ。MPEGに対してこれは10ビットの整数である。H.261のみに対しては、5つのLSBが用いられる。MSBは常にゼロである。
1	1	1	1	0	0	1	0	0	TIME_CODE
1	r	r	r	r	h	h	h	h	MPEGタイム_コード
1	r	r	d	d	d	d	d	d	d; ドロップフレームフラグ
1	r	r	s	s	s	s	s	s	h; 時間を指定する5ビット整数
0	r	r	p	p	p	p	p	p	m; 分を指定する6ビット整数
									s; 秒を指定する6ビット整数
									p; ピクチャを指定する6ビット整数

表A.32 空間デコーダ及び時間デコーダにおいて実現されるトークン(8/9)

[0485]

[表11]

E	7	6	5	4	3	2	1	0	記 述
1	0	0	0	1	1	0	1	1	USER DATA JPEG
0	v	v	v	v	v	v	v	v	このトークンは、ビデオデマックスに接続するDATAトークンが、ユーザデータを含むことを通知する。 A.11.3 (スタートコードからトークンへの交換) 及び (ユーザ及び拡張データの受取り) 参照。 JPEG動作期間中、8ビットフィールドはJPEGマーカ値を持つ。これによりユーザデータのクラスが識別可能となる。
0	0	0	0	1	1	0	1	1	USER DATA MPEG
									このトークンは、ビデオデマックスに、接続するDATAトークンがユーザデータを含むことを通知する。 A.11.3 (スタートコードからトークンへの交換) 及び A.14.6 (ユーザ及び拡張データの受取り) 参照。
1	1	1	1	0	1	1	0	1	VBV BUFFER SIZE
1	r	r	r	r	r	r	s	s	s; MPEGの定義による1つの10ビット整数。
0	s	s	s	s	s	s	s	s	
1	1	1	1	0	1	1	1	0	VBV DELAY
1	b	b	b	b	b	b	b	b	b; MPEGの定義による1つの16ビット整数。
0	b	b	b	b	b	b	b	b	
1	1	1	1	1	1	1	0	1	VERTICAL MBS
1	r	r	r	v	v	v	v	v	v; マクロブロックにおけるピクチャの垂直サイズを指示する1つの13ビット整数。
0	v	v	v	v	v	v	v	v	
1	1	1	1	1	0	0	1	1	VERTICAL SIZE
1	v	v	v	v	v	v	v	v	v; 画素におけるピクチャの垂直サイズを指示する1つの16ビット整数。これは任意の整数値であって差支えない。
0	v	v	v	v	v	v	v	v	

表A.3.2 空間デコード及び時間デコードにおいて実現されるトークン(9/9)

A. 3. 5 「トークンにおいて送信される番号」

る。MPEG、及びH. 261の場合には、相互関係は

A. 3. 5. 1 「成分識別番号」

30 非常に簡単である。

本発明に基づく成分ID番号は、色成分を指定する2ビ

【0486】

ット整数である。この2ビットフィールドは、一般に、

【表12】

データトークンにおけるヘッダの一部として配置され

成 分ID	MPEG又はH. 261色成分
0	輝度 (Y)
1	青色差信号 (Cb/U)
2	赤色差信号 (Cr/V)
3	一切使用せず

表A.3.3 MPEG及びH. 261用成分ID

JPGでは使用できる色成分は制限されないで、J
PEGの場合には、条件は更に複雑である。デコードチ
ップは、各スキャンにおいて最大4種までの異なる色成
分を可能にする。色成分の仕様がデコードに到着するに
つれて、IDは順次に割当てられる。

【0487】 A. 3. 5. 2 「水平および垂直サンプ
リング数」

4種の色成の各々に関しては、マクロブロックにおいて

40 水平および垂直に配列されるブロックの数に関する仕様
がある。この仕様には、2ビットの整数を含むブロック
数より1つだけ少ない2ビット整数が含まれる。例え
ば、MPEG (或いはH. 261) における4:2:0
クロマサンプリング (図45) の成分IDの配列を表1
3に示す。

【0488】

【表13】

成分ID	水平 サンプリング 番号	ブロック 幅	垂直 サンプリング 番号	ブロック 高さ
0	1	2	1	2
1	0	1	0	1
2	0	1	0	1
3	使用せず	使用せず	使用せず	使用せず

表A.3.4 4:2:0/MPEG用サンプリング番号

JPEG及び4:2:2クロマサンプリングに関して、合には、JPEGは、そのマクロブロックに対して2:成分IDへの成分割当は、用途によって異なる。セクシ 10 1:1構造を必要とする。

ンA. 3. 5. 1参照。

【0490】

【0489】注記：4:2:2データを処理している場

【表14】

成分ID	水平 サンプリング 番号	ブロック 幅	垂直 サンプリング 番号	ブロック 高さ
Y	1	2	0	1
U	0	1	0	1
V	0	1	0	1

表A.3.5 4:2:2/JPEG用サンプリング番号

A. 3. 6 「特殊トークンフォーマット」

本発明に基づき、例えば、データトークン及びQUANTテーブルトークンは、デコーダチップ・セット内におけるそれらの「拡張フォーム」に使用される。拡張フォームにおいて、トークンは、いくつかのデータを含む。データトークンの場合には、この種トークンは、コード化データまたはピクセルデータを含むことができる。QUANTテーブルトークンの場合には、これらのトークンは、量子化器表情報を含む。

20 【0491】更に、これらのトークンの「非拡張形」

は、本発明においては「空である」として定義される。このトークンフォーマットは、同一トークンの拡張バージョンによって続いて満たされる場所をトークンストリーム内に提供する。このフォーマットは、主としてエンコーダに適用可能であり、従って、これについては、これ以上詳しく説明しない。

【0492】

【表15】

トークン名	MPEG	JPEG	H.261
BIT RATE	/	.	
BROKEN CLOSED	/	.	
CODING STANDARD	/	/	/
COMPONENT NAME		/	
CONSTRAINED	/		
DATA	/	/	/
DEFINE MAX SAMPLING	/	/	/
DEFINE SAMPLING	/	/	/
DHT MARKER		/	
DNL MARKER		/	
DQT MARKER		/	
DRI MARKER		/	

表A.3.6 異なる規格用トークン(1/2)

【0493】

【表16】

トークン名	MPEG	JPEG	H.261
EXTENSION DATA	/	/	
FIELD INFO			
FLUSH	/	/	/
GROUP START	/	/	/
HORIZONTAL MBS	/	/	/
HORIZONTAL SIZE	/	/	/
JPEG TABLE SELECT		/	
MAX COMP ID	/	/	/
MPEG DCH TABLE	/		
MPEG TABLE SELECT	/		
MVD BACKWARDS	/		
MVD FORWARDS	/		/
NULL	/	/	/
PEL ASPECT	/		
PICTURE END	/	/	/
PICTURE RATE	/		
PICTURE START	/	/	/
PICTURE TYPE	/	/	/
PREDICTION MODE	/	/	/
QUANT SCALE	/		/
QUANT TABLE	/	/	
SEQUENCE END	/	/	
SEQUENCE START	/	/	/
SLICE START	/	/	/
TEMPORAL REFERENCE	/		/
TIME CODE	/		
USER DATA	/	/	
VBV BUFFER SIZE	/		
VBV DELAY	/		
VERTICAL MBS	/	/	/
VERTICAL SIZE	/	/	/

表A.3.6 異なる規格用トークン(2/2)

A. 3. 7 「異なる規格に対するトークンの使用」 30
各規格は、本発明に基づいて定義されたトークンの異なる部分集合を使用する。

【0494】セクションA. 4 「2線インターフェース」

A. 4. 1 「2線インターフェース、及びトークンポート」

チップ・セットにおいては、情報の流れを制御するために、簡単な2線有効/受入れプロトコルが用いられる。送信側及び受信側においてクロック発生準備が整っていることが観察される場合に、データは、ブロック間にお 40
いて単に転送されるだけである。

【0495】1) データ転送

2) 受信側の準備が整っていない

3) 送信側の準備が整っていない

送信側の準備が整っていない場合には(上記の3)送信側の準備が整っていない場合には、受信側の入力待たなければならない。受信側の準備が整っていない場合には(上記の2)受信側の準備が整っていない場合に

は、受信側によって受け入れられる時まで送信側はその出力に同じデータを供給し続ける。

【0496】トークン情報がブロック間において転送される場合には、ブロック間の2線インターフェースはトークンポートと呼ばれる。

【0497】A. 4. 2 「使用場所」

本発明に基づくデコーダチップ・セットは、3つのチップを接続するために2線インターフェースを使う。更に、空間デコーダへのコード化されたデータ入力も、同様に、2線インターフェースである。

【0498】A. 4. 3 「バス信号」

2線インターフェースによって転送されるデータワードの幅は、関係インターフェースの必要性に応じて変化する(図44「8ビットより広いインターフェース上のトークン」参照)。例えば、12ビットの係数は、僅か9ビットを出力するだけで、離散逆コサイン変換器(IDCT)に入力される。

【0499】

【表17】

インターフェース	データ幅(bits)
空間デコーダへのコード化データ入力	8
空間デコーダの出力ポート	9
時間デコーダの入力ポート	9
時間デコーダの出力ポート	8
イメージフォーマット部の入力ポート	8

表A.4.1 2線インターフェースデータの幅

データ信号に加えて、2線インターフェース経由で送られる他の3つの信号がある。

【0500】*有効信号(valid)

*受入れ信号(accept)

*拡張信号(extension)

A. 4. 3. 1 「拡張信号」

拡張信号は、すでに述べたトークン拡張ビットに対応する。

【0501】A. 4. 4 「設計上の考察事項」

2線インターフェースは、短距離、チップ間逐次通信用である。

【0502】デコーダチップは、チップの間のPCBトラックの長さを最小限にするために、相互に近接して配置されなければならない。場所的に可能な限り、トラックの長さは、25mm以下に保持されなければならない

い。PCBトラックキャパシタンスは、最小限度に保持されなければならない。

10 【0503】クロックの分散配置は、チップ間のクロックスルーを最小限にするように設計されなければならない。何等かのクロックスルーがある場合には、「受信チップ」が、「送信チップ」より前にクロックを見るように、配列しなければならない。

【0504】2線インターフェースを介して通信する全てのチップは、同一デジタル電源から作動しなければならない。

【0505】A. 4. 5 「インターフェースタイミング」

【0506】

【表18】

No.	特 性	30MHz		単位	注記
		最小	最大		
1	入力信号準備時間	5		ns	
2	入力信号保持時間	0		ns	
3	出力信号ドライブ時間		23	ns	
4	出力信号保持時間	2		ns	

a; 数値は仮の値であり、変更可能である。

b; 最大信号ローディングは、20. Fである。

表A.4.2 2線インターフェースのタイミング

注記：図47は、システムデマックスチップと、メインデコーダクロックから作動する空間デコーダのコード化データポートとの間の2線インターフェースを示す。この2線インターフェースはデコーダクロックに対して非同期であっても差し支えないコード化データクロックから作動可能なので、これはオプションである。セクションA. 10. 5、「コード化されたデータクロック」を参照されたい。同様に、イメージフォーマット部

のディスプレイインターフェースは、メインデコーダクロックに対して非同期であるクロックから作動可能であ

る。

【0507】A. 4. 6 「信号レベル」

2線インターフェースは、COMS入力、及び出力を使用する。VIHminはVDDの約70%であり、VILmaxはVDDの約30%である。表19に示す値は、VDDが最悪の場合におけるVIH、及びVILの値である。VDD=5.0±0.25V。

【0508】

【表19】

記号	パラメータ	最小	最大	単位
V_{IH}	入力ロジック "1" 電圧	3.68	$V_{DD}-0.5$	V
V_{IL}	入力ロジック "0" 電圧	GND, 0.5	1.43	V
V_{OH}	出力ロジック "1" 電圧	$V_{DD}-0.1$		V ^a
		$V_{DD}-0.4$		V ^b
V_{OL}	出力ロジック "0" 電圧	-	0.1	V ^c
			0.4	V ^d
I_{IH}	入力漏れ電流		±10	μA

表A.4.3 DC電気特性

a: $I_{OH} \leq 1\text{mA}$ b: $I_{OH} \leq 4\text{mA}$ c: $I_{OL} \leq 1\text{mA}$ d: $I_{OL} \leq 4\text{mA}$

A. 4. 7 「制御クロック」

一般に、2線インターフェースを介して転送を制御するクロックはチップのデコードクロックである。空間デコードへのコード化データポート入力は例外である。これは、符号化クロックによって制御される。クロック信号についてはここで更に説明することとする。

【0509】セクション A. 5 「DRAMインターフェース」

A. 5. 1 「DRAMインターフェース」

1つの単一高性能、構成可能DRAMインターフェースは、各々の動画デコードチップにおいて使用される。一

般に、各チップ上のDRAMインターフェースは実質的に同じである。ただし、インターフェースは、チャネル優先権を扱う方法において相互に異なる。インターフェースは、各々のデコードチップによって使われるDRAMを直接ドライブするように設計されている。一般に、外部の論理、バッファ、またはコンポーネントは、DRAMインターフェースを大抵のシステムにおけるDRAMに接続するためには不必要である。

【0510】A. 5. 2 「インターフェース信号」

【0511】

【表20】

信号名	入力/出力	記 述
DRAM_data[31:0]	入/出	幅32ビットのDRAMデータバス。このバスは幅16又は8ビットに、定期的に減縮可能である。セクションA. 5. 8参照
DRAM_addr[10:0]	出	幅22ビットのDRAMインターフェースアドレスは、この幅11ビットのバスを介して時間多重化される。
RAS	出	DRAM行アドレスストローブ信号
CAS[3:0]	出	信号は、インターフェースのデータバスのバイト毎に供給される。全てのCAS信号は同時にドライブされる。
WE	出	DRAMライト可能化信号
OE	出	DRAM出力可能化信号
DRAM_enable	入	この入力信号は、lowの場合、インターフェース上の全ての出力信号を高インピーダンスにする。 注記：オンチップデータ記憶はDRAMインターフェースが高インピーダンスの場合、停止されない。従って、DRAM_イネイブルがlowの場合には、チップがアクセスを試みた場合にエラーが生じる。

表A. 5. 1 DRAMインターフェース信号

本発明に基づき、インターフェースは、次に示す2つの方法において構成可能である。

【0512】*インターフェースの詳細タイミングは、様々な異なるタイプのDRAMを収容するように構成可

能である。

【0513】*DRAMインターフェースの「幅」は、異なるアプリケーションにおけるコスト/性能トレードオフを提供するように構成可能である。

【0514】A. 5. 3 「DRAMインターフェースの構成」

一般に、DRAMインターフェースと関連したレジスタの3つのグループがある、即ち、インターフェースタイミング・コンフィギュレーション・レジスタ、バス・コンフィギュレーション・レジスタ、及びリフレッシュ・コンフィギュレーション・レジスタである。リフレッシュ・コンフィギュレーション・レジスタ表23に示すレジスタは最後に構成されなければならない。

【0515】A. 5. 3. 1 「リセット後の条件」
リセットの後において、本発明に基づくDRAMインターフェースは、1組のデフォルトタイミングパラメータ（最も遅い作動モードに対応する）を用いて動作を開始する。初めに、DRAMインターフェイスは、リフレッシュサイクル（他の全ての転送を除く）を連続的に実行する。これは、値がリフレッシュ・インターバルに記入されるまで、継続する。次に、DRAMインターフェースは、リフレッシュサイクル間の他のタイプの転送を行うことができる。

【0516】A. 5. 3. 2 「バスコンフィギュレーション」

バスコンフィギュレーション22におけるレジスタは、インターフェースによるデータ転送が試みられていない場合に限り、行われるべきである。インターフェースは、リセットのすぐ後、値がリフレッシュインターバルに記入される以前に、この条件に置かれる。インターフェースは後で、転送が試みられていない場合に限り、必要に応じて、再構成可能である。時間デコーダチップアクセスレジスタ（A. 18. 3. 1）、及び空間デコーダ・バッファマネージャアクセス・レジスタ（A. 13. 1. 1）参照。

【0517】A. 5. 3. 3 「インターフェースタイミングの構成」

本発明に基づき、インターフェースタイミング構成情報の修正は、インターフェースタイミングアクセスレジスタによって制御される。このレジスタに1を記入すると、インターフェースタイミングレジスタ21に示す）が修正可能になる。インターフェースタイミングアクセス=1である間に、DRAMインターフェースは、その以前のコンフィギュレーションにより作動し続ける。1を記入した後で、任意のインターフェースタイミングレ

ジスタに記入する以前において、インターフェースタイミングアクセスから1が読戻し可能になるまで、ユーザーは待たねばならない。

【0518】コンフィギュレーションが完了すると、インターフェースタイミングアクセスに0が記入されなければならない。次に、新しいコンフィギュレーションは、DRAMインターフェースへ転送される。

【0519】A. 5. 3. 4 「リフレッシュコンフィギュレーション」

10 本発明にかかるDRAMインターフェースのリフレッシュインターバルは、リセットに続いて一度だけ構成可能である。リフレッシュ・インターバルが構成されるまで、インターフェースは、リフレッシュサイクルを連続的に実行する。これは、あらゆる他のデータ転送を阻止する。リフレッシュ・インターバルに値が記入された後で、データ転送はスタート可能である。

【0520】当該技術分野においては周知であるように、DRAMは、一般に、最初に電源を入れた後で、100マイクロ秒と500マイクロ秒との間に「ポーズ（休止）」を必要とし、そして、正常作動が可能になる前に、多数のリフレッシュサイクルが後続する。従って、リフレッシュ・インターバルに値を記入する前に、これらのDRAMスタート必要条件を満足させなければならない。

【0521】A. 5. 3. 5 「コンフィギュレーションレジスタへのリードアクセス」

本発明の全てのDRAMインターフェースレジスタは読み取り可能である。

【0522】A. 5. 4 「インターフェースタイミング（チック）」

DRAMインターフェースタイミングは、デバイス（デコーダクロック）の入力クロックレートの4倍のレートにおいてランするクロックから得られるこのクロックは、オンチップPLLによって生成される。

【0523】説明を簡潔にするために、この高速クロックの周期をチックと称する。

【0524】A. 5. 5 「インターフェースレジスタ」

【0525】

【表21】

40

レジスタ名	ビット	初期値	説 明
interface_ timings_ access	1 bit rw	0	この書き換えレジスタは、DRAMインタフェースタイミング 設定レジスタへのアクセスを可能にする。書き換えレジスタは、この レジスタが0を保持している期間中は書き換えされない。 このレジスタに0が記入された後で、DRAMインタフェースは タイミング設定レジスタ内の所定の値の適用を開始する。
access_start_ length	5 bit rw	0	アクセススタートの長さを指定する。使用可能な最小値は4 (4 デックを意味する) である。0は最大値32デックを指定す る。
transfer_ cycle_ length	4 bit rw	0	転送ページ読み取り又は書き込みサイクルの長さを指定する (デック 数を表す)。使用可能な最小値は4 (4デックを意味する)。0 は最大値16デックを指定する。
refresh_ cycle_ length	4 bit rw	0	リフレッシュサイクルの長さをデック数で指定する。使用できる 最小値は4である (4デックを意味する)。0は最大値16デッ クを指定する。
RAS_falling	4 bit rw	0	RASがフォールするアドレススタート開始後のデック数を指定 する。使用できる最小値は4である (4デックを意味する)。0 は最大値16デックを指定する。
CAS_falling	4 bit rw	8	リードサイクル、ライトサイクル又はRASがフォールするアク セススタート開始後のデック数を指定する。使用できる最小値は 1である (1デックを意味する)。0は最大値16デックを指定 する。

図A. 5. 2 インタフェースタイミングコンフィギュレーションレジスタ

【0526】

【表22】

レジスタ名	ビット	初期値	説 明
DRAM_data_width	2 bit rw	0	DRAMインタフェースデータバス DRAMデー タ[31:0] で使用するビット数を指定する。 A. 5. 8参照
row_address_bits	2 bit rw	0	DRAMインタフェースアドレスバスの列アドレス使 用に使用されるビット数を指定する。 A. 5. 10参照
DRAM_enable	1 bit rw	1	このレジスタに0を記入すると、DRAMインタフ ェースを高インピーダンス状態に強制する。DRAM _enableレジスタがlowであるか、又は、レジスタ に0が記入済みである場合には、レジスタから0が 読み出される。
CAS_strength	3 bit rw	6	これらのビットレジスタはDRAMインタフェース 信号の出力ドライバ強度を指定する。これによって 様々なロードに対してインタフェースが最適可 能になる。 A. 5. 13参照
RAS_strength			
addr_strength			
DRAM_data_strength			
OEWE_strength			

図A. 5. 3 インタフェースバスコンフィギュレーションレジスタ

A. 5. 6 「インターフェースオペレーション」
DRAMインターフェースは高速ページモードを用い
る。3つの異なるタイプのアクセスがサポートされる。

【0527】* 読取り (リード)

* 書込み (ライト)

* リフレッシュ

各々のリード又はライトアクセスは、1から64バイト
までの1つのバーストを1つの単一DRAMページアド
レスへ転送する。リード及びライト転送は、1つの単一

50 アクセス内において混合されことなく、そして 連続

する各アクセスは、新しいDRAMページへのランダム
アクセスとして扱われる。

【0528】

【表23】

レジスタ名	ビット 状態	記 述
refresh interval	8 bit rw	この値は、16デコードクロックサイクル期間にお けるリフレッシュサイクル間インターバルを指定する。 レンジ1...255における値は構成可能である。値 0は、リセット後に自動的にロードされ、そして、 有効なリフレッシュインターバルが構成されるまで、 DRAMインタフェースがリフレッシュサイクルを 連続的に実行するよう強制する。リフレッシュイン タバルは、各リセット後においてただ1度に限り構 成されることを推薦する。
no refresh	1 bit rw	値1をこのレジスタに記入すると、あらゆるリフレ ッシュサイクルの実行を防止する。

表A.5.4 リフレッシュコンフィギュレーションレジスタ

A. 5. 7 「アクセスの構造」

各アクセスは2つの部分を有する。

【0529】* アクセススタート

* データ転送

本発明においては、各アクセスはアクセススタートで始
まり、そして、1つ又は複数のデータ転送サイクルが後
続する。更に、アクセススタート及びデータ転送サイク
ルの両方のリード、ライト、及びリフレッシュ変形（バ
リエーション）がある。

【0530】特定のアクセスに対する最後のデータ転送
が完成すると、インタフェースは、そのデフォルト状
態に入り（A. 5. 7. 3参照）、そして、新しいアク
セスのスタート準備が整うまで、この状態に留まる。最

20 後のアクセスが終了して新しいアクセスの開始準備が整
った場合、直ちに新しいアクセスが始まる。

【0531】A. 5. 7. 1 「アクセススタート」

アクセススタートは、リード又はライト転送のためのペ
ージアドレスを提供し、そして、ある程度の初期信号条
件を確立する。本発明に基づき、3つの異なるアクセス
スタートがある。

【0532】* 読取りのスタート

* 書込みのスタート

* リフレッシュのスタート

30 【0533】

【表24】

No.	特 性	最小	最大	単位	注記
5	レジスタRAS_フォーリングによりセッ トされるRASプリチャージ期間	4	15	チック	
6	レジスタページ_スタート_レンジによ りセットされるアクセススタート期間	4	32		
7	レジスタCAS_フォーリングによってセ ットされるCASプリチャージ長さ	1	15		
8	レジスタトランスファ_サイクル_レン グスによってセットされる迅速ページリー ド又はライトサイクル長さ	4	15		
9	レジスタリフレッシュ_サイクルによっ てセットされるリフレッシュサイクル長さ	4	15		

a: この値は、RASリフレッシュが起きる以前におけるCASを保証するために、
RAS falling未満でなくてはならない。

表A.5.5 DRAMインタフェースタイミングパラメータ

各々の場合に、RAS及び行アドレスのタイミングは、
レジスタRAS_フォーリング、及びページスタートレン
グスによって制御される。OE、及びDRAMデータ

【31:0】の状態は、前のデータ転送の終了からRA
Sが降下するまで秘事される。アクセススタートの3つ
の異なるタイプは、RASが降下した場合に、これらが

50

OE及びDRAMデータ[31:0]をドライブする方法が異なるに過ぎない。参照図52。

【0534】A. 5. 7. 2 「データ転送」

本発明においては、異なるタイプのデータ転送サイクルがある。

【0535】* 高速ページ読取りサイクル

* 高速ページ後期書込みサイクル

* リフレッシュサイクル

リフレッシュのスタートは、1つの単一リフレッシュサイクルによってのみ後続可能である。読み取り（或いは、書き込み）のスタートは、1つ又は複数の高速ページ読取り（或いは、書き込み）サイクルによって後続可能である。読み取りサイクルのスタートに際して、CASはハイにドライブされ、そして、新しい列アドレスがドライブされる。

【0536】更に、早期書込みサイクルが用いられる。WEは、最初の書込み転送のスタートに際してローにドライブされ、そして、最後の書き込み転送のエンドまで、ローに留どまる。出力データは、アドレスによってドライブされる。

【0537】RASリフレッシュサイクルの前のCASはリフレッシュサイクルのスタートによって開始される

DRAM_data width	
0*	DRAM_data[31:24] ^a の幅8ビットデータバス
1	DRAM_data[31:16] ^a の幅16ビットデータバス
2	DRAM_data[31] の幅32ビットデータバス

a: リセット後デフォルト

b: 使用されない信号は、高インピーダンスに保持される。

表A.56 DRAM data widthの構成

A. 5. 9 「行アドレス幅」

行アドレスを提供するために24ビット内部アドレスの中間区分から取られるビット数は、ローアドレスビット

row address bits	列アドレスの幅
1	DRAM_addr[9:0]の10ビット
2	DRAM_addr[10:0]の11ビット

表A.57 row address bitsの構成

A. 5. 10 「アドレスビット」

オンチップ24ビットアドレスが生成される。列及び行アドレスを形成するためにこのアドレスをどのように使用するかは、データバスの幅、及び行アドレス用に選定されたビット数に依存する。コンフィギュレーションによっては、全ての内部アドレスビットを使用することが

ので、リフレッシュサイクル期間中には、インターフェース信号は活動しない。リフレッシュサイクルの目的は、DRAMによって必要とされる最小RASロー期間に適合することである。

【0538】A. 5. 7. 3 「インターフェースのデフォルト状態」

本発明におけるインターフェース信号は、アクセスのエンドにおいてデフォルト状態に入る。

【0539】RAS、CAS、及びWEがハイ。

【0540】* データ、及びOEは、それらの前の状態に留まる。

【0541】* addrは、安定した状態を維持する。

【0542】A. 5. 8 「データバス幅」

2ビットレジスタ、DRAMデータwidthは、DRAMインターフェースのデータバスの幅が構成されることを可能にする。これは、小さいピクチャフォーマットで使用する場合、DRAMコストを最小限にすることを可能にする。

【0543】

【表25】

30 ・レジスタによって構成される。

【0544】

【表26】

許されないので、「隠れビット」が作成される。

【0545】同様に、行アドレスは、アドレスの中央部分から抽出される。従って、これにより、DRAMが自然にリフレッシュされる速度が最大になる。

【0546】

【表27】

列 アドレス 幅	列アドレス 変換 内部→外部	データ バス 幅	カラムアドレス変換 内部→外部	
9	[14:6] → [8:0]	8	[19:15] → [10:6]	[5:0] → [5:0]
		16	[20:15] → [10:5]	[5:1] → [4:0]
		32	[21:15] → [10:4]	[5:2] → [3:0]
10	[15:6] → [9:0]	8	[19:16] → [10:5]	[5:0] → [5:0]
		16	[20:15] → [10:5]	[5:1] → [4:0]
		32	[21:16] → [10:4]	[5:2] → [3:0]
11	[16:6] → [10:0]	8	[19:17] → [10:6]	[5:0] → [5:0]
		16	[20:17] → [10:5]	[5:1] → [4:0]
		32	[21:17] → [10:4]	[5:2] → [3:0]

表A.5.8 内部アドレスと外部アドレス間のマッピング

A. 5. 10. 1 「低順位行アドレスビット」

行（カラム）アドレスの最下位の4から6までのビットは、最大64バイトまでの高速ページモード転送用アドレスを提供するために使われる。これらの転送を制御するために要求されるアドレスビット数は、データバスの幅に依存する（A. 5. 8参照）。

【0547】A. 5. 10. 2 「より多くのDRAMバンクにアクセスするための行アドレス復号化」

DRAMのただ1つの単一バンクが用いられる場合には、使用される行アドレスの幅は、用いられるDRAMのタイプに依存する。一般に1つの単一DRAMバンクによって供給可能であるよりもより多くのメモリーを必要とするアプリケーションの場合には、より広い行アドレスを構成することが可能であり、従って、1つの単一DRAMバンクを選定するために、幾つかの行アドレスビットを復号化することが可能である。

【0548】注記：行アドレスは、内部アドレスの中央から抽出される。DRAMのバンクを選定するために、行アドレスの幾つかのビットが復号化される場合には、これらの「バンク選定ビット」がとることのできる全ての値がDRAMの1つのバンクを選定しなければならない。そうでない場合には、アドレススペース内に孔が残される。

【0549】A. 5. 11 「DRAMインターフェースの作動可能化」

本発明においては、DRAMインターフェース上の全ての出力信号を高インピーダンスにするためには2つの方法がある。即ち、DRAMenableレジスタをセットする方法及びDRAMenable信号による方法である。DRAMインターフェース上のドライバを作動させるためには、これらのレジスタ及び信号は、両方共に、ロジック1でなければならない。どちらかがローであれば、インターフェースは高インピーダンスになる。

【0550】注記：DRAMインターフェースが高インピーダンスである場合には、オンチップデータ処理は終了させられない。従って、チップがDRAMへのアクセスを試みた場合にはエラーが発生し、同時に、インター

フェースは高インピーダンスである。

【0551】本発明に基づき、DRAMインターフェースを高インピーダンスにする能力は、他のデバイスをテストするか、或いは、空間デコーダ（又は、時間デコーダ）が使用中でない場合、空間デコーダ（又は、時間デコーダ）によって制御されるDRAMを使用するために備えられる。正常作動期間中において、他のデバイスによるメモリーの共有が可能であるようには意図されていない。

【0552】A. 5. 12 「リフレッシュ」

ノーリフレッシュ・レジスタに記入することにより無能化しない限り、DRAMインターフェースは、レジスタリフレッシュ・インターバルによって決定されたインターバルにおいて／RAS／リフレッシュサイクルの以前の／CAS／を使用し、DRAMを自動的にリフレッシュする。ここで、／信号名／は信号の反転信号を示す。

【0553】リフレッシュ・インターバルにおける値は、16デコーダクロックサイクルの周期内におけるリフレッシュサイクル間のインターバル（間隔）を指定する。レンジ1. 255における値は構成可能である。値0は、リセットの後で自動的にロードされ、そして、（いったん、作動可能化されれば）有効なリフレッシュインターバルが構成されるまで、DRAMインターフェイスが、リフレッシュサイクルを連続的に実行することを強制する。リフレッシュ・インターバルは、各リセットの後で、ただ1度だけ構成されることが推奨される。

【0554】／リセット／が表明されると、同時に、DRAMインターフェースは、DRAMをリフレッシュすることが不可能となる。ただし、デコーダチップによって必要とされるリセット時間は十分に短く、その結果、これらのデコーダチップをリセットし、そして、次に、DRAMの内容が腐敗する以前に、DRAMインターフェースを再構成することが可能であるはずである。

【0555】A. 5. 13 「信号強度(strength)」

DRAMインターフェースの出力のドライブ強度は、3ビットレジスタCASストレングス、RASストレングス、addtストレングス、DRAMデータストレング

ス、及びOEWEストレングスを用いてユーザーにより構成可能である。この3ビット値のMSBは、高速或いは低速エッジレート of the いずれかを選択する。より重要度の低い2つのビットは、異なる負荷キャパシタンスのための出力を構成する。

【0556】リセット後のデフォルト強度は6であり、

強度値	ド ラ イ ブ 特 性
0	Approx. 4 ns into 6 pf load
1	Approx. 4 ns into 12 pf load
2	Approx. 4 ns into 24 pf load
3	Approx. 4 ns into 48 pf load
4	Approx. 2 ns into 6 pf load
5	Approx. 2 ns into 12 pf load
6*	Approx. 2 ns into 24 pf load
7	Approx. 2 ns into 48 pf load

a: リセット後デフォルト

表A.5.9 出力強度コンフィギュレーション

出力がドライブしつつある負荷に対して出力が適切に構成された場合、その出力は、表32から表35までに指定されたAC電気特性に適合する。適切に構成された場合、各出力は、その負荷に概略マッチし、従って、信号過渡の後に最小のオーバシュートが発生する。

【0558】A. 5. 14 「電気仕様」

そして、これは、24 pFにより負荷した場合、GNDとVDDとの間の信号をドライブするために約10 nsを必要とするような出力を構成する。

【0557】

【表28】

このセクションに提示される全ての情報は、単に本発明の1実施例を示すに過ぎず、一例として記載されているものであって、必ずしも制限的意味を持たない。

【0559】

【表29】

記 号	パラメータ	最小	最大	単位
V _{DD}	GNDに対する供給電圧	0.5	6.5	V
V _{IN}	任意のピン上の入力電圧	GND, 0.5	V _{DD} +0.5	V
T _A	作動温度	-40	+85	°C
T _S	保存温度	-55	+150	°C

表A.5.10 最大定格

表29は、実施例に関する最大定格を示すに過ぎない。 30 ならない。

特にこの実施例のみに関して、動作の信頼性を保証するためには、この表に示す値未満の応力を使用しなければ

【0560】

【表30】

記 号	パラメータ	最小	最大	単位
V _{DD}	GNDに対する供給電圧	4.75	5.25	V
GND	接地	0	0	V
V _{IH}	入力ロジック "1" 電圧	2.0	V _{DD} +0.5	V
V _{IL}	入力ロジック "0" 電圧	GND, 0.5	0.8	V
T _A	作動温度	0	70	°C

a: TBA線形 ft/min 横方向空流による。

表A.5.11 DC作動条件

【0561】

【表31】

記 号	パラメータ	最 小	最 大	単 位
V_{OL}	出力ロジック "0" 電圧		0.4	V*
V_{OH}	出力ロジック "1" 電圧	2.8		V
I_O	出力電流	± 100		μA^*
I_{OL}	出力オフ状態漏れ電流	± 20		μA
I_{IL}	入力漏れ電流	± 10		μA
I_{DD}	RMS電源電流		500	mA
C_{IN}	入力キャパシタンス		5	pF
C_{OUT}	出力/IOキャパシタンス		5	pF

- a ; ACパラメータは、測定レベルとして $V_{OLMAX} = 0.8$ Vを用いて指定される。
- b ; これは、インタフェースの安定状態ドライブ能力である。過渡電流は、遙かに大きい可能性がある。

表A.5.12 DC電気特性

A. 5. 14. 1 「AC特性」
[0562]

【表32】

No.	パラメータ	最 小	最 大	単 位	注記
10	サイクル時間	-2	+2	ns	
11	サイクル時間	-2	+2	ns	
12	高パルス	-5	+2	ns	
13	低パルス	-11	+2	ns	
14	サイクル時間	-8	+2	ns	

- a ; 当該技術分野の通常習熟者であれば当然理解できるように信号ドライブ強度は、その負荷に応じて適当に構成されなければならない。

表A.5.13 1ストロブとの公称値との差

[0563]

【表33】

No.	パラメータ	最 小	最 大	単 位	注記
15	ストロブからストロブまでの遅延	-3	+3	ns	
16	low保持時間	-13	+3	ns	
17	ストロブからストロブまでのプリチャージ、例えば t_{CRP} , t_{RCS} , t_{RCH} , t_{RRH} , t_{RPC} 例えば t_{CP} のような広いDRAMにおける又は例えば t_{RPC} のような \overline{RAS} 上昇とで \overline{AS} 降下との間における任意の2つの \overline{CAS} 信号の間の \overline{CAS} プリチャージパルス。	-9	+3	ns	
		-5	+2	ns	
18	不能化前プリチャージ	-12	+3	ns	

- a ; 2つの信号のドライブ強度は、それらの負荷に応じて適当に構成されなければならない。

表A.5.14 2ストロブ間の公称値との差

[0564]

40 【表34】

No.	パラメータ	最 小	最 大	単 位	注記
19	セットアップ時間	-12	+3	ns	
20	保持時間	-12	+3	ns	
21	アドレスアクセス時間	-12	+3	ns	
22	ストロブ後の次の有効	-12	+3	ns	

- a ; バス及びストロブのドライブ強度は、それらの負荷に応じて適当に構成されなければならない。

表A.5.15 バスとストロブ間の公称値との差

[0565]

50 【表35】

No.	パラメータ	最 小	最 大	単 位	注 記
23	CAS信号開始から立ち上がり前のリードデータセットアップ時間	0		ns	
24	CAS信号開始からhighになった後のリードデータ保持時間	0		ns	

表A.5.16 バスとストローブ間の公称値との差

DRAMから読み出す場合、DRAMインターフェース 【0566】
 は、/CAS/信号の上昇につれて、DRAMデータ 【表36】
 [31:0] をサンプルする。

10

パラメータ		パラメータ		パラメータ	
name	number	name	number	name	number
tPC	10	tRSH	16	tRHCP	18
				tCPRH	
tRC	11	tCSH		tASR	19
tRP	12	tRWL		tASC	
tCP		tCWL		tDS	
tCPN		tRAC		tRAH	20
tRAS	13	tOAC/tOE		tCAH	
tCAS		tCHR		tDH	
tCAC		tCRP	17	tAR	
tWP		tRCS		tAA	21
tRASP		tRCH		tRAL	
tRASC		tRRH		tRAD	22
tACP/tCPA	14	tRPC			
tRCD	15	tCP			
tCSR		tRPC			

表A.5.17 “規格” DRAMパラメータ名とタイミングパラメータ番号との間の相互参照

セクション A. 6 「マイクロプロセッサインターフェース (MPI)」

標準バイト幅マイクロプロセッサインターフェース (MPI) は、動画デコーダチップ・セットにおける全てのチップに用いられる。ただし、当該技術分野における通常の熟練者であれば、他の幅のマイクロプロセッサイン

ターフェースであっても同様に使用可能であることが理解されるはずである。MPI は、様々なデコーダチップクロックと同期的に作動する。

【0567】 A. 6. 1 「MPI 信号」

【0568】

【表37】

信号名	入力/ 出力	記 述
$\overline{\text{enable}}[1:0]$	入力	2動作化lowチップ動作可能化。MPIを介してアクセス可能であるためには両方ともlowでなくてはならない。
$\overline{\text{rv}}$	入力	highはデバイスがビデオチップから値を読取ことを望むことを示す。この信号はチップが動作化されている期間中、安定していなければならない。
$\text{addr}[n:0]$	入力	アドレスは、チップのメモリマップにおける2 ⁿ ロケーションのうちの1つを指定する。この信号は、チップが動作化されている期間中、安定していなければならない。
$\text{data}[7:0]$	出力	幅8ビットのデータI/Oポート。いずれかの動作可能化信号がhighであればこれらのピンは高インピーダンスである。
$\overline{\text{irq}}$	出力	動作化low、戻コレクタ、書き込みリクエスト信号

表A.6.1 MPIインターフェース信号

A. 6. 2 「MPI電気仕様」

【表38】

【0569】

20

記号	パラメータ	最小	最大	単位
V_{DD}	GNDに対する供給電圧	0.5	6.5	V
V_{IN}	任意のピン上の入力電圧	GND, 0.5	$V_{DD}+0.5$	V
T_A	動作温度	-40	+85	°C
T_S	保存温度	-55	+150	°C

表A.6.2 絶対最大定格

【0570】

【表39】

記号	パラメータ	最小	最大	単位
V_{DD}	GNDに対する供給電圧	4.75	5.25	V
GND	接地	0	0	V
V_{IH}	入力ロジック "1" 電圧	2.0	$V_{DD}+0.5$	V ^a
V_{IL}	入力ロジック "0" 電圧	GND, 0.5	0.8	V ^(a)
T_A	動作温度	0	70	°C ^b

a: AC入力パラメータは、1.4 V測定レベルにおいて測定される。

b: TBA線形 ft/min 横方向空気流による。

表A.6.3 DC動作条件

【0571】

【表40】

記号	パラメータ	最小	最大	単位
V_{OL}	出力ロジック "0" 電圧		0.4	V
V_{OLX}	開コレクタ出力ロジック "0" 電圧		0.4	V ^a
V_{OH}	出力ロジック "1" 電圧	2.4		V
I_O	出力電流	± 100		μA^b
I_{OX}	開コレクタ出力電流	4.0	8.0	mA ^c
I_{OZ}	出力オフ状態漏れ電流		± 20	μA
I_{IN}	入力漏れ電流		± 10	μA
I_{CO}	RMS電源電流		500	mA
C_{IN}	入力キャパシタンス		5	pF
C_{OUT}	出力/IOキャパシタンス		5	pF

a: $1.0 \leq I_{OXmax}$

b: これは、インタフェースの安定状態ドライブ能力である。

過渡電流は、遙かに大きい可能性がある。

c: 表明されている場合、開コレクタ I_{OZ} 出力は、100 Ω 以下のインピーダンスにより、プルダウンする。

表 A. 6.4 DC電気特性

A. 6. 2. 1 「AC特性」

【表 4 1】

【0572】

No.	特 性	最 小	最 大	単 位	注 記
25	可能化 low 期間	100		ns	
26	可能化 high 期間	60		ns	
27	chip可能化へのアドレス又はセットアップ	0		ns	
28	chip可能化からのアドレス又はホールド	0		ns	
29	出力turn-on 時間	20		ns	
30	リードデータアクセス時間	70	ns	(b)	
31	リードデータホールド時間	5		ns	
32	リードデータturn-off時間		20		

表 A. 6.5 マイクロプロセッサ インタフェース読取りタイミング

a: この例においては、サイクルをスタートさせる $\overline{enable}[0]$ 、及びそれを終了させる $\overline{enable}[1]$ の送定は任意である。これらの信号は同じ状態である。

b: アクセスタイムは各データ[7:0] に関して50. Fの最大負荷に対して指定される。負荷が大きくなるとアクセスタイム増大があり得る。

【0573】

【表 4 2】

No.	特 性	最 小	最 大	単 位	注 記
33	ライトデータセットアップ時間	15		ns	*
34	ライトデータホールド時間	0		ns	

a: この例では、サイクルをスタートさせる $\overline{enable}[0]$ 、及びそれを終了させる $\overline{enable}[1]$ の送定は、任意である。これらの信号は同じ状態である。

表 A. 6.6 マイクロプロセッサインタフェース書き込みタイミング

A. 6. 3 「割込み」

本発明において、「イベント」は、ユーザーが観察しようとする可能性のあるオンチップ条件を示すために使われる用語である。イベントは、エラーを示すことが可能であるか、或いは、ユーザーのソフトウェアにとって有益である場合もあり得る。

【0574】各割込み、または「イベント」と関連した2つの単一ビットレジスタがある。これらは、条件イベ

ントレジスタ及び条件マスクレジスタである。

【0575】A. 6. 3. 1 「条件イベントレジスタ」

条件イベントレジスタは、1ビット読み/書きレジスタであり、その値は、回路内に発生する条件によって1にセットされる。条件が単に一時的であり、そして、現在では過ぎ去ってしまった場合であってもレジスタは1にセットされる。従って、レジスタは、ユーザーのリブ

ウェアによってリセットされるまで、1にセットされたままであることを保証される（或いは、チップ全体がリセットされる）。

【0576】*レジスタは、値1を記入することによりゼロにセットされる。

【0577】*レジスタにゼロを記入することにより、レジスタを変更されない状態に留める。

【0578】*レジスタは、この条件がも一度発生する以前にユーザーソフトウェアによってゼロにセットされなければならない。

【0579】*レジスタは、リセットに関してゼロまでリセットされるであろう。

【0580】A. 6. 3. 2 「条件マスクレジスタ」
条件マスクレジスタは、対応する条件イベントレジスタがセットされ場合、割込みリクエストの生成を可能にする1ビット読み/書きレジスタである。既に条件イベントがセットされている場合、条件マスクに1が記入されると、割込みリクエストが即座に発行される。

【0581】*値1は割込みを可能にする。

【0582】*レジスタは、リセットに際してゼロにクリアされる。

【0583】別途表明されていない限り、ブロックは、割込みリクエストを生成した後で動作を中止し、そして、条件イベント又は条件マスクレジスタのいずれかがクリアされた後で動作を再開する。

【0584】A. 6. 3. 3 「イベント及びマスクビット」

イベントビット、及びマスクビットは、メモリマップ内の連続したバイト内の対応するビット位置に常にグループとして配置される表61、表141参照）。これは、どのイベントが割込みを生成したかを識別するために、割込みサービスソフトウェアが、マスクとしてのマスクレジスタから読み取られた値を、イベントレジスタ内の値に対するマスクとして使用することを可能にする。

【0585】A. 6. 3. 4 「チップイベント、及びマスク」

各チップは、チップ上のイベント活動を要約する1つの単一グローバルイベントビットを持つ。チップイベントレジスタは、それらのマスクビットに1を持つ全てのオンチップイベントのORを提供する。

【0586】チップマスクビットにおける1は、チップによる割込みの生成を可能にする。チップマスクビットにおける0は、あらゆるオンチップイベントが割込みリクエストを生成することを防止する。

【0587】チップイベントへの1から0までの記入は影響しない。全てのイベント（どちらかのマスクビットに1を記入することによって作動化される）がクリアされた場合に限り、チップはクリアされる。

【0588】A. 6. 3. 5 「irq信号」

チップイベントビット及びチップイベントマスクの両方

がセットされている場合に、/irq/信号が表明される。

【0589】/irq/信号は、オフチッププルアップ抵抗器を必要とするアクティブローのオープンコレクタ出力である。活動中である場合、/irq/出力は、100オーム以下のインピーダンスによってプルダウンされる。

【0590】殆どの用途の場合に、約4キロオームのプルアップ抵抗器が適当であることが理解されるはずである。

【0591】A. 6. 4 「レジスタへのアクセス」
A. 6. 4. 1 「アクセスを可能にするストップ回路」

本発明における大部分のレジスタは、レジスタが関連しているブロックが停止されている場合に限り、修正が可能である。従って、レジスタのグループは、一般に、アクセスレジスタと関連している。

【0592】アクセスレジスタにおける値0は、当該アクセスレジスタと関連したレジスタのグループが修正されてはならないことを示す。アクセスレジスタに1を記入すると、ブロックが停止されることを要求する。ただし、ブロックは即座に停止しなくても差し支えなく、そして、当該ブロックのアクセスレジスタは、当該ブロックが停止するまで、値0を保持する。

【0593】従って、ユーザーソフトウェアは（アクセスを要求するために1を記入した後で）、アクセスレジスタから1が読み取られるまで、待つはずである。アクセスレジスタは0にセットされた状態において、ユーザーがコンフィギュレーションレジスタに値を記入すると、結果は不定である。

【0594】A. 6. 4. 2 「整数を保持するレジスタ」

メモリマップ内の任意のバイトの最下位ビットは、信号データ[0]と関連しているビットである。

【0595】8ビットより大きい整数値を保持するレジスタは、メモリマップ内の2つ又は4ついずれか連続したバイト位置に互って分割される。バイトの順序は、図64に示すように「ビッグエンディアン」である。ただし、しかしながら仮定は、オーダについて行われたいバイトがマルチバイトレジスタに記入される順序に関しては仮定は一切設定されていない。

【0596】符号付き整数を保持するレジスタ内の未使用ビットを除くメモリマップ内の未使用ビットは、読み取りに際して0を返す。この場合、レジスタ内の最上位のビットは符号拡張される。例えば、符号付きの12ビットレジスタは、16ビットメモリマップ位置（2バイト）を満たすために、符号拡張される。符号無し12ビット整数を保持する16ビットメモリマップ位置は、その最上位ビットから0を返す。

【0597】A. 6. 4. 3 「キーホールされたアド

レス位置」

本発明において、アクセス頻度の小さいメモリマップ位置は、「キーホール」の背後に配置される。「キーホール」は、関連した2つのレジスタを持つ、即ち、キーホールアドレスレジスタ、及びキーホールデータレジスタである。

【0598】キーホールアドレスは、拡張されたアドレススペース内の位置を指定する。キーホールデータレジスタに対する読取り、または書込み動作は、キーホールアドレスレジスタによって指定された位置にアクセスする。

【0599】関連キーホールアドレスレジスタは、キーホールデータレジスタにアクセスした後で、インクリメントする。拡張されたアドレススペース内でのランダムアクセスは、新しいバリュウ値を各アクセスに対するキーホールアドレスレジスタに新しい値を記入することによってのみ可能である。

【0600】本発明に基づくチップは、1つよりも多い「キーホールされた」メモリマップを備えていても差し支えない。異なるキーホールの間にはインタラクションはない。

【0601】A. 6. 5 「特殊レジスタ」

A. 6. 5. 1 「未使用レジスタ」

「使用せず」と記載されたレジスタ又はビットは、本デバイスの実現に使用されなかったメモリマップ内の位置である。一般に、これらの位置からは値0が読み取り可能である。これらの位置に0を記入しても影響しない。

【0602】当該技術分野における通常の習熟者であれば理解出来るように、将来における変形製品との互換性をこの種の製品に維持させるためには、ユーザーのソフトウェアが、未使用の位置から読み取られた値に依存してはならないことが推奨される。同様に、デバイスの構成に際しては、この種の位置は、回避するか、或いは、値0にセットするべきである。

【0603】A. 6. 5. 2 「予約済みレジスタ」

信号名	入力/出力	記 述
axded clock	入力	このクロックは、空間デコーダのコード化データポートに対するデータ転送を制御する。オンチップのこのクロックは、コード化データバッファに到達するまで、コード化データの処理を制御する。
decoder clock	入力	デコーダクロックは、空間デコーダの処理機能の大部分を制御する。更に、デコーダクロックは、その出力ポートを介して空間デコーダからのデータ転送を制御する。

表A.7.1 空間デコーダクロック

A. 7. 2 「時間デコーダクロック信号」

時間デコーダはただ1つのクロック入力を持つ。

同様に、本発明において「予約済み」と記載されたレジスタ又はビットは、デバイスの作動態様にはドキュメント化された影響は及ぼさない。

【0604】A. 6. 5. 3 「テストレジスタ」

更に、「テストレジスタ」と記載されたレジスタ又はビットは、当該デバイスの被テスト性の様々な態様を制御する。従って、これらのレジスタは、当該デバイスの標準的な用途には使用されず、そして、標準的なデバイスコンフィギュレーション及び制御ソフトウェアによってアクセスされる必要はない。

【0605】セクション A. 7 「クロック」

本発明に基づき、多くの異なるクロックは、動画デコーダシステムにおいて識別可能である。クロックの例を、図65に示す。

【0606】データは、動画デコーダチップ・セット内の異なるクロックレジームの間を供給するにつれて、それぞれ新しいクロックに再同期化（オンチップ）する。本発明において、あらゆる入力クロックの最大周波数は30MHzである。ただし、当該技術分野における通常に習熟者であれば、30MHzより大きい周波数を含む他の周波数であっても同様に使用可能であることは、当然理解できるはずである。各チップにおいて、マイクロプロセッサインターフェース（MPI）は、チップクロックに対して非同期的に作動する。更に、イメージフォーマッティング部は、復号化された動画のピクチャレートに同期した低い周波数のオーディオクロックを生成することが出来る。従って、このクロックは、オーディオ／動画同期を供給するために使用できる。

【0607】A. 7. 1 「空間デコーダクロック信号」

空間デコーダは、2つの異なる（そして、非同期的である可能性のある）クロック入力を持つ。

【0608】

【表43】

【0609】

【表44】

信号名	入力/出力	記 述
decoder clock	入力	デコードクロックは、時間デコードにおける全ての処理機能を制御する。更に、デコードクロックは、その入力ポート及び出力ポートを介して、時間デコードに対するデータ転送を制御する。

表A.7.2 時間デコードクロック

A. 7. 3 「電気仕様」

【表 4 5】

【0610】

10

No.	特 性	30 MHz		単位	注記
		最小	最大		
35	クロック期間	33		ns	
36	クロックhigh期間	13		ns	
37	クロックlow期間	13		ns	

表A.7.3 入力クロック必要条件

【0611】

【表 4 6】

記号	パラメータ	最小	最大	単位
V_{IH}	入力ロジック "1" 電圧	3.68	$V_{DD} \pm 0.5$	V
V_{IL}	入力ロジック "0" 電圧	$GND \pm 0.5$	1.43	V
I_{oz}	入力漏れ電流		± 10	μA

表A.7.4 クロック入力条件

A. 7. 3. 1 「CMOSレベル」

クロック入力信号はCMOS入力である。 V_{IHmin} はVDDの約70%であり、そして、 V_{ILmax} はVDDの約30%である。表46に示す値は、それぞれVDDの最悪条件における V_{IH} 及び V_{IL} の値である。 $V_{DD} = 5.0 \pm 0.25V$

A. 7. 3. 2 「クロックの安定性」

本発明において、DRAMインターフェース及びチップ間インターフェースをドライブするために用いられるクロックは、入力クロック信号から得られる。これらのインターフェースのためのタイミング仕様は、入力クロックタイミングが $\pm 100ps$ の範囲内で安定しているものと仮定する。

【0612】セクション A. 8 「JTAG」

回路ボードの設置密度が高くなるにつれて、例えば「ベッドオブナイル (bed-of-nails)」技法を用いた回路内テストのような従来の方法によって部品間の接続を検査することはますます困難になる。アクセス問題の解決および方法論の標準化の試みとして、ジョイントテストアクショングループ (JTAG) が形成された。このグループの仕事の集大成が現在では規格1149.1としてIEEEによって採用されている「標準テストアクセスポート及び境界走査アーキテクチャ」である。空間デコード及び時間デコードは、この規格に適合する。

【0613】規格は、デバイス上の各デジタル信号ピンを直列接続する境界スキャンチェーンを利用する。テスト回路は、正常作動状態においては即応型 (透明) で

ある、しかし、テストモードにおいて境界スキャンチェーンはテストパターンをシフトイン可能にし、そして、デバイスのピンに供給可能にする。JTAGデバイスへの入力において回路ボードに現れる結果的な信号は、比較的簡単なテスト装置によって走査およびチェックされることが可能である。この方法により、回路ボード上のロジックの領域と同様に部品間接続はテスト可能である。

【0614】全てのJTAGオペレーションは5つのピンを有するテストアクセスポート (TAP) を介して実施される。 t_{rst} (テストリセット) ピンは、デバイスがテストモードにおいてはパワーアップしないことを保証するためにJTAG回路をリセットする。 t_{ck} (テストクロック) ピンは、クロック直列テストパターンを、 t_{di} (テストデータ入力) ピンへクロックし、そして、 t_{do} (テストデータ出力) ピンからクロックするために用いられる。最後に、JTAG回路の操作モードは、該当するビットのシーケンスを t_{ms} (テストモードセレクト) ピンへクロックすることによりセットされる。

【0615】JTAG規格は、チップメーカーの裁量において付加的特徴を提供するために拡張可能である。空間デコード及び時間デコードには、3つのJTAG必須命令を含めて9つのユーザインストラクション (命令) がある。追加命令 (エクストラインストラクション) は、特定の程度の内部デバイステストの実施を可能にし、そして、付加的な外部テストの融通性を提供する。例え

30

40

50

ば、全てのデバイス出力は、簡単なJTAGシーケンスによってフロートするように作成可能である。

【0616】利用可能な機能およびJTAGポートの使用方法の詳細については、個別のJTAGアプリケーションノートを参照されたい。

【0617】A. 8. 1 「非JTAGシステムにおけるJTAGピンの接続」

【0618】

【表47】

信号	方向	記 述
trst	入力	このピンは、内部プルアップを持つが、JTAG機能を使用していない場合であっても、パワーアップに際してはlowでなくてはならない。これは、共通trstとチップリセットピンresetを接続することによって達成される。
tdi	入力	これらのピンは、内部プルアップを持ち、JTAG回路が使用されていない場合には接続しないままでも差支えない。
tdo	出力	このピンは、プルアップを持たず、JTAG回路が使用されていない場合には、接地しなければならない。
tx	入力	JTAG走査作動期間を除き、高インピーダンス。JTAGが使用されていない場合には、このピンは接続しないままであっても差支えない。

表A.8.1 JTAG入力接続法

A. 8. 2 「IEEE 1149. 1への適合レベル」

下記の条項が注記されるが、全ての規則が適合する。

【0619】

A. 8. 2. 1 「規則」

20 【表48】

規則	記 述
3.1.1(b)	trstピンを備える。
3.5.1(b)	全ての公的規定に対して保証済み (IEEE, 1149.1, 5.2.1(c)参照)
5.2.1(c)	全ての公的規定に対して保証済み。接つかの個人的規定に対しては、TDOピンは、状態(Capture-DR, Exit1-DR, Exit2-DR及びPause-DRの任意の状態)の期間中、作動化されていても差支えない。
5.3.1(a)	電源on-resetは、trstピンを用いて達成される。
6.2.1(e, f)	BYPASS命令用コードは、Test-Logic-Reset状態においてロードされる。
7.1.1(d)	割当てられない命令コードはBYPASSと等価である。
7.2.1(c)	デバイスIDレジスタ無し。

表A.8.2 JTAG規則

【0620】

【表49】

規則	記 述
7.8.1(b)	単一ステップ作動時には、システムクロックの外部的制御が必要である。
7.8.1(c...)	RUNBIST 機能無し。
7.11.1(c...)	IDCODE-命令無し。
7.12.1(c...)	USERCODE 命令無し。
8.1.1(b)	デバイス識別レジスタ無し。
8.2.1(c)	全ての公的命令に対して保証済みである。個人的命令コードがロードされている場合には、tdi から tdoまでの見掛け長さは、ある種状況下においては変化可能である。
8.3.1(d, i)	全ての公的命令に対して保証済みである。個人的命令コードがロードされている場合には、データは tckの立ち上がりエッジ以外の時点においてデータのロードが可能である。
10.4.1(e)	INTEST期間中、システムクロックピンは外部的に制御されなければならない。
10.5.1(c)	INTEST期間中、出力ピンは、tdi を介してシフトインされたデータによって制御される。

表A.8.2 JTAG規則

A. 8. 2. 2 「勧告条件」

【表50】

【0621】

勧告	記 述
3.2.1(b)	tck は高インピーダンスCMOS入力である。
3.3.1(c)	tms は高インピーダンスプルアップを持つ。
3.6.1(d)	(チップ使用)
3.7.1(a)	(チップ使用)
6.1.1(e)	SAMPLE/PRELOAD命令コードは、Capture-IR期間中にクリアされる。
7.2.1(f)	INTEST命令がサポートされる。
7.7.1(g)	EXTEST期間中、システム出力ピンにはゼロがロードされる。
7.7.2(b)	全てのシステム出力が高インピーダンスにセットされても差支えない
7.8.1(f)	INTEST期間中、システム出力ピンにゼロがロードされる。
8.1.1(d, e)	設計指定テストデータレジスタは公的にアクセス不可能である。

表A.8.3 適合勧告条件

【0622】

【表51】

勧告	記 述
10.4.1(f)	EXTEST期間中、システムクロックピンからオンチップロジックにドライブされる信号は外部供給された信号である。

表A.8.4 実現されない勧告条件

A. 8. 2. 3 「許可条件」

【表52】

【0623】

許可	記 述
3.2.1(c)	全ての公的命令に対して保証済み
6.1.1(f)	インストラクションレジスタは、設計指向情報を捕捉するために用いられない。
7.2.1(g)	幾つかの公的追加命令が供給される。
7.3.1(a)	幾つかの私的命令コードが等価でられる。
7.3.1(c)	(規則?) この種のインストラクションコードは文書化されていない。
7.4.1(f)	追加コードは、BYPASSに対して同様に作用する。
10.1.1(f)	各出力ピンは、それぞれ固有の3状態制御を持つ。
10.2.1(b)	並列ラッチが提供される。
10.3.1(f, g)	EXTEST期間中、入力ピンは、tck を介してシフトインされたデータによって制御される。
10.6.1(d, e)	3状態セルは、Test-logic-Reset状態において、作動不能化に強制されない。

表A.8.5 適合許可条件

セクション A. 9 「空間デコーダ」

- * 30MHz、オペレーション
- * デコーダ-MPEG、JPEG、及びH. 261
- * コード化データレートは25Mb/sまで
- * 動画データレートは21MB/sまで
- * 柔軟なクロマサンプリングフォーマット
- * 全JPEG基底線復号化
- * グルーレスDRAMインターフェース
- * 単一+5V電源
- * 208ピンPQFPパッケージ
- * 最大消費電力2.5W
- * 独立したコード化データ及びデコーダクロック

* 標準ページモードDRAMを使用

- 40 空間デコーダは、様々なJPEG、MPEG、及びH. 261ピクチャ及び動画復号化アプリケーションにおいて使用するための構成可能なVLSIデコーダチップである。

【0624】オフチップDRAMを使用しない最小のコンフィギュレーションにおいて、空間デコーダは、1つの単一チップ、高速JPEGデコーダである。DRAMを加えることにより、空間デコーダがJPEG符号化動画を復号化することを可能にする。720×480、30Hz、4:2:2「JPEG動画」が、リアルタイムに復号化可能である。

173

【0625】時間デコーダを用いることにより、空間デコーダは、H. 261、及びMPEG（同様に、JPEG）を復号化するために使用できる。704×480、30Hz、4:2:0MPEG動画が復号化可能である。

【0626】前述の値は単に一例に過ぎず、例として示したものであって、必ずしも制限的意図を持つものでな

174

く、本発明に基づく1実施例のための典型的な値であることを再度注記しておく。従って、当該技術分野における通常の習熟者であれば、他の値、及び／又は、を使用可能であることを理解するはずである。

【0627】A. 9. 1 「空間デコーダ信号」

【0628】

【表53】

信号名	I/O	ピンナンバー	説 明
codec_clock	I	182	コード化データ又はトーンを空間デコーダに供給するために使用されるコード化されたデータポート、 セクションA.10.1 又はA.4.1参照
coded_data[7:0]	I	172, 171, 169, 168, 167, 166, 164, 163	
coded_extn	I	174	
coded_valid	I	162	
coded_accept	O	181	
byte_acde	I	176	
enable[1:0]	I	126, 127	
rv	I	125	マイクロプロセッサインタフェース (MPI) セクションA.6.1参照
addr[6:0]	I	136, 135, 133, 132, 131, 130, 128	
data[7:0]	O	152, 151, 149, 147, 145, 143, 141, 140	
trq	O	154	
DRAM_data[31:0]	I/O	15, 17, 19, 20, 22, 25, 27, 30, 31, 33, 35, 36, 39, 42, 44, 47, 49, 57, 59, 61, 63, 66, 68, 70, 72, 74, 76, 79, 81, 83, 84, 85	
DRAM_addr[10:0]	O	184, 186, 188, 189, 192, 193, 195, 197, 199, 200, 203	
RAS	O	11	
CAS[3:0]	O	2, 4, 6, 8	DRAMインタフェース セクションA.5.2参照
WE	O	12	
OE	O	204	
DRAM_enable	I	112	
out_data[8:0]	O	88, 89, 90, 92, 93, 94, 95, 97, 98	
out_extn	O	87	
out_valid	O	99	
out_accept	I	100	出力ポート セクションA.5.2参照
tcz	I	115	
tcl	I	116	
tde	O	120	
tms	I	117	
trst	I	121	
decoder_clock	I	177	JTAGポート セクションA.8参照
reset	I	160	
			メインデコーダクロック セクションA.7参照
			リセット

図A.9.1 空間デコーダ信号

【0629】

【表54】

図号名	I/O	ピン No.	記 述
lph0ish	I	122	オーバーライド=1であれば、lph0ish及びlph1ishは、オンチップ2相クロック用入力である。正常動作に対しては、オーバーライド=0にセットすること。lph0ish及びlph1ishは接続される（GND又はV _{DD} に接続される）。
lph1ish	I	123	
override	I	110	
ch0test	I	111	正常動作に対しては、チップセット=0にセットすること。
lloop	I	114	正常動作中、GND又はV _{DD} に接続すること。
ramtest	I	109	ramtestが1であれば、オンチップRAMのテストが動作可能化される。正常動作に対しては、ramtest=0にセットすること。
pllselect	I	178	pllselect=0であれば、オンチップ位相クロックされたループは動作不能化される。
tl	I	180	テスト動作中において、DRAMインタフェースによって必要とされる2つのクロック。正常動作中は、GNDまたはV _{DD} に接続すること。
td	I	179	
pdout	O	207	これら2つのピンは、位相クロックループ用外部フィルタに対する接続部である。
pdin	I	206	

表A.9.2 空間デコーダテスト図号

【0630】

【表55】

図号名	ピン	図号名	ピン	図号名	ピン	図号名	ピン
dc	228	nc	155	nc	104	nc	52
test_0in	207	nc	153	nc	103	nc	51
test_0in	206	17F	154	nc	102	nc	50
GND	205	nc	153	VDD	101	GRAM_data[15]	49
OE	204	data[7]	152	ext_accout1	100	nc	48
DRAM_addr[0]	203	data[6]	151	ext_valid	99	DRAM_data[16]	47
VDD	202	nc	150	ext_data[0]	98	nc	46
nc	201	data[5]	149	ext_data[1]	97	GND	45
DRAM_addr[1]	200	nc	148	GND	96	DRAM_data[17]	44
DRAM_addr[2]	199	data[4]	147	ext_data[2]	95	nc	43
GND	198	GND	146	ext_data[3]	94	DRAM_data[18]	42
DRAM_addr[3]	197	data[3]	145	ext_data[4]	93	VDD	41
nc	196	nc	144	ext_data[5]	92	nc	40
DRAM_addr[4]	195	data[2]	143	VDD	91	DRAM_data[19]	39
VDD	194	nc	142	ext_data[6]	90	DRAM_data[20]	38
DRAM_addr[5]	193	data[1]	141	ext_data[7]	89	nc	37
DRAM_addr[6]	192	data[0]	140	ext_data[8]	88	GND	36
nc	191	nc	139	ext_extin	87	DRAM_data[21]	35
GND	190	VDD	138	GND	86	nc	34
DRAM_addr[7]	189	nc	137	DRAM_data[0]	85	DRAM_data[22]	33
DRAM_addr[8]	188	addr[0]	136	DRAM_data[1]	84	VDD	32
VDD	187	addr[5]	135	DRAM_data[2]	83	DRAM_data[23]	31
DRAM_addr[9]	186	GND	134	VDD	82	DRAM_data[24]	30
nc	185	addr[4]	133	DRAM_data[3]	81	nc	29
DRAM_addr[10]	184	addr[3]	132	nc	80	GND	28
GND	183	addr[2]	131	DRAM_data[4]	79	DRAM_data[25]	27
code_clock	182	addr[1]	130	GND	78	nc	26
VDD	181	VDD	129	nc	77	DRAM_data[26]	25
test_0in	180	addr[0]	128	DRAM_data[5]	76	nc	24
test_0in	179	data[0]	127	nc	75	VDD	23
test_0in	178	data[1]	126	DRAM_data[6]	74	DRAM_data[27]	22
decoder_clock	177	rd	125	VDD	73	nc	21
byte_mode	176	GND	124	DRAM_data[7]	72	DRAM_data[28]	20
GND	175	test_0in	123	nc	71	DRAM_data[29]	19
code_extin	174	test_0in	122	DRAM_data[8]	70	GND	18

表A.9.3 空間デコーダピンの割り振り

【0631】

【表56】

信号名	ピン	信号名	ピン	信号名	ピン	信号名	ピン
nc	208	nc	156	nc	104	nc	52
test pin	207	nc	155	nc	103	nc	51
test oia	206	ifs	154	nc	102	nc	50
GND	205	nc	153	VDD	101	GRAM_data[15]	49
OE	204	data[7]	152	out_accept	100	nc	48
DRAM_addr[0]	203	data[6]	151	out_valid	99	DRAM_data[16]	47
VDD	202	nc	150	out_data[0]	98	nc	46
nc	201	data[5]	149	out_data[1]	97	GND	45
DRAM_addr[1]	200	nc	148	GND	96	DRAM_data[17]	44
DRAM_addr[2]	199	data[4]	147	out_data[2]	95	nc	43
GND	198	GND	146	out_data[3]	94	DRAM_data[18]	42
DRAM_addr[3]	197	data[3]	145	out_data[4]	93	VDD	41
nc	196	nc	144	out_data[5]	92	nc	40
DRAM_addr[4]	195	data[2]	143	VDD	91	DRAM_data[19]	39
VDD	194	nc	142	out_data[6]	90	DRAM_data[20]	38
DRAM_addr[5]	193	data[1]	141	out_data[7]	89	nc	37
DRAM_addr[6]	192	data[0]	140	out_data[8]	88	GND	36
nc	191	nc	139	out_exta	87	DRAM_data[21]	35
GND	190	VDD	138	GND	86	nc	34
DRAM_addr[7]	189	nc	137	DRAM_data[9]	85	DRAM_data[22]	33
DRAM_addr[8]	188	addr[6]	136	DRAM_data[10]	84	VDD	32
VDD	187	addr[5]	135	DRAM_data[11]	83	DRAM_data[23]	31
DRAM_addr[9]	186	GND	134	VDD	82	DRAM_data[24]	30
nc	185	addr[4]	133	DRAM_data[12]	81	nc	29
DRAM_addr[10]	184	addr[3]	132	nc	80	GND	28
GND	183	addr[2]	131	DRAM_data[13]	79	DRAM_data[25]	27
coded_clock	182	addr[1]	130	GND	78	nc	26
VDD	181	VDD	129	nc	77	DRAM_data[26]	25
test pin	180	addr[0]	128	DRAM_data[14]	76	nc	24
test pin	179	enable[0]	127	nc	75	VDD	23
test pin	178	enable[1]	126	DRAM_data[15]	74	DRAM_data[27]	22
decoder_clock	177	r#	125	VDD	73	nc	21
byte_mode	176	GND	124	DRAM_data[16]	72	DRAM_data[28]	20
GND	175	test oia	123	nc	71	DRAM_data[29]	19
coded_exta	174	test pin	122	DRAM_data[17]	70	GND	18

表A.9.3 信号デコーダピンの配列(2/3)

【0632】

【表57】

信号名	ピン	信号名	ピン	信号名	ピン	信号名	ピン
nc	173	ifs	121	GND	69	DRAM_data[30]	17
coded_data[7]	172	ids	120	DRAM_data[18]	68	nc	16
coded_data[8]	171	nc	119	nc	67	DRAM_data[31]	15
VDD	170	VDD	118	DRAM_data[19]	66	VDD	14
coded_data[5]	169	ims	117	VDD	65	nc	13
coded_data[4]	168	idi	116	nc	64	WE	12
coded_data[3]	167	ick	115	DRAM_data[11]	63	RAS	11
coded_data[2]	166	test oia	114	nc	62	nc	10
GND	165	GND	113	DRAM_data[12]	61	GND	9
coded_data[1]	164	DRAM_enable	112	GND	60	CAS[0]	8
coded_data[0]	163	test pin	111	DRAM_data[13]	59	nc	7
coded_valid	162	test pin	110	nc	58	CAS[1]	6
coded_accept	161	test oia	109	DRAM_data[14]	57	VDD	5
reset	160	nc	108	VDD	56	CAS[2]	4
VDD	159	nc	107	nc	55	nc	3
nc	158	nc	106	nc	54	CAS[3]	2
nc	157	nc	105	nc	53	nc	1

表A.9.3 信号デコーダピンの配列(3/3)

A. 9. 1. 1 「nc」非接続ピン

にしておかねばならない。

表57においてncとラベルの付いたピンは現在使われていないピンを表す。これらのピンは、接続しない状態

【0633】 A. 9. 1. 2 「VDD、及びGNDピ

ン

当該技術分野における通常の習熟者であれば理解されるように、装備されている全てのV_{DD}及びGNDピンは、それぞれ該当する電源に接続されなければならない。全てのV_{DD}及びGNDピンが正しく使用されない限り、デバイスの正しい作動は保証不可能である。

【0634】A. 9. 1. 3 「正常作動のためのテスト

ピンの接続」

空間デコーダの9つのピンは、内部テスト用として予約済みである。

【0635】

【表58】

ピン ナンバー	接 続
	正常作動に対しては、GNDに接続すること。
	正常作動に対しては、V _{DD} に接続すること。
	正常作動に対しては開回路状態のままにしておくこと

表A.9.4 デフォルトテストピンの接続

A. 9. 1. 4 「正常作動のための」TAGピン」

プ」

セクションA. 8. 1参照。

【0637】

【0636】A. 9. 2 「空間デコーダメモリマップ

【表59】

アドレス(16進)	レジスタ名	表参照
0x00...0x03	読み込みサービス領域	A.9.6
0x04...0x07	入力回路レジスタ	A.9.7
0x08...0x0F	スタートコード検出器レジスタ	
0x10...0x15	バッファスタートアップ制御レジスタ	A.9.8
0x16...0x17	使用されず	
0x18...0x23	DRAMインタフェース構成レジスタ	A.9.9
0x24...0x26	バッファマネージャアクセス及びキーホールレジスタ	A.9.10
0x27	使用されず	
0x28...0x2F	ハフマンデコーダレジスタ	A.9.13
0x30...0x39	逆量子化レジスタ	A.9.14
0x3A...0x3B	使用されず	
0x3C	予約済み	
0x3D...0x3F	使用されず	
0x40...0x7F	テストレジスタ	

表A.9.5 空間デコーダメモリマップの概要

【0638】

【表60】

181

182

アドレス (16進)	ビット No.	登録名	参照頁
0x00	7	chip_event CED_EVENT_0	
	6	使用せず	
	5	illegal_length_count_event SCD_ILLEGAL_LENGTH_COUNT	
	4	予約済み、既取り又は0 SCD_JPEG_OVERLAPPING_START	
	3	overlapping_start_event SCD_NON_JPEG_OVERLAPPING_START	
	2	unrecognised_start_event SCD_UNRECOGNISED_START	
	1	stop_after_picture_event SCD_STOP_AFTER_PICTURE	
	0	non_aligned_start_event SCD_NON_ALIGNED_START	
0x01	7	chip_mask CED_MASK_0	
	6	使用せず	
	5	illegal_length_count_mask	
	4	予約済み、この場所への書き込み0 SCD_JPEG_OVERLAPPING_START	
	3	non_jpeg_overlapping_start_mask	
	2	unrecognised_start_mask	
	1	stop_after_picture_mask	
	0	non_aligned_start_mask	
0x02	7	idct_too_few_event IDCT_DEFF_NUM	
	6	idct_too_many_event IDCT_SUPER_NUM	
	5	accept_enable_event BS_STREAM_END_EVENT	
	4	target_met_event BS_TARGET_MET_EVENT	
	3	counter_flushed_too_early_event BS_FLUSH_BEFORE_TARGET_MET_EVENT	
	2	counter_flushed_event BS_FLUSH_EVENT	

表A.9.6 刻込みサービス領域レジスタ(1/2)

[0639]

[表61]

アドレス (16進)	ビット No.	登録名	参照頁
0x02	1	parser_event DEMUX_EVENT	
	0	huffman_event HUFFMAN_EVENT	
0x03	7	idct_too_few_mask	
	6	idct_too_many_mask	
	5	accept_enable_mask	
	4	target_met_mask	
	3	counter_flushed_too_early_mask	
	2	counter_flushed_mask	
	1	parser_mask	
	0	huffman_mask	

表A.9.6 刻込みサービス領域レジスタ(2/2)

[0640]

[表62]

183

184

アドレス (16進)	ビット No.	登録名	参照頁
0x04	7	coded_busy	
	6	enable_spl_input	
	5	coded_extn	
	4:0	使用せず	
0x05	7:0	coded_data	
0x06	7:0	使用せず	
0x07	7:0	使用せず	
0x08	7:1	使用せず	
	0	start_code_detector_access also input_circuit_access CED_SCD_ACCESS	
0x09	7:4	使用せず CED_SCD_CONTROL	
	3	stop_after_picture	
	2	discard_extension_data	
	1	discard_user_data	
	0	ignore_non_aligned	
0x0A	7:5	使用せず CED_SCD_STATUS	
	4	insert_sequence_start	
	3	discard_all_data	
	2:0	start_code_search	

表A.9.7 スタートコード検出器及び入力回路レジスタ

【0641】

【表63】

アドレス (16進)	ビット No.	登録名	参照頁
0x08	7:0	Test register length_count	
0x0C	7:0		
0x0D	7:2	使用せず	
	1:0	start_code_detector_coding_standard	
0x0E	7:0	start_value	
0x0F	7:4	使用せず	
	3:0	picture_number	

表A.9.7 スタートコード検出器及び入力回路レジスタ(2/2)

【0642】

【表64】

185

186

アドレス (16 進)	ビット No.	登 録 名	参照頁
0x10	7:1	使用せず	
	0	startup_access CED_BS_ACCESS	
0x11	7:3	使用せず	
	2:0	bit_count_prescale CED_BS_PRESCALE	
0x12	7:0	bit_count_target CED_BS_TARGET	
0x13	7:0	bit_count CED_BS_COUNT	
0x14	7:1	使用せず	
	0	offchip_queue CED_BS_QUEUE	
0x15	7:1	使用せず	
	0	enable_stream CED_BS_ENABLE_NXT_STM	

表A.9.8 バッファスタートアップレジスタ

【0643】

【表65】

アドレス (16 進)	ビット No.	登 録 名	参照頁
0x18	7:5	使用せず	
	4:0	page_start_length CED_IT_PAGE_START_LENGTH	
0x19	7:4	使用せず	
	3:0	read_cycle_length	
0x1A	7:4	使用せず	
	3:0	write_cycle_length	

表A.9.9 DRAMインタフェースコンフィギュレーションレジスタ(1/2)

【0644】

【表66】

アドレス (16進)	ビット No.	登録名	参照頁
0x1B	7:4	使用せず	
	3:0	refresh_cycle_length	
0x1C	7:4	使用せず	
	3:0	CAS_falling	
0x1D	7:4	使用せず	
	3:0	RAS_falling	
0x1E	7:1	使用せず	
	0	interface_tuning_access	
0x1F	7:0	refresh_interval	
0x20	7	使用せず	
	6:4	DRAM_addr_strength[2:0]	
	3:1	CAS_strength[2:0]	
	0	RAS_strength[2]	
0x21	7:6	RAS_strength[1:0]	
	5:3	OEW_strength[2:0]	
	2:0	DRAM_data_strength[2:0]	
0x22	7	バッド使用他ACCESSビットではないか? usedCED_DRAM_CONFIGURE	
	6	zero_buffers	
	5	DRAM_enable	
	4	no_refresh	
	3:2	row_address_bits[1:0]	
	1:0	DRAM_data_width[1:0]	
0x23	7:0	Test registers CED_PLL_RES_CONFIG	

表A.9.9 DRAMインタフェースコンフィギュレーションレジスタ(2/2)

【0645】

【表67】

アドレス (16進)	ビット No.	登録名	参照頁
0x24	7:1	使用せず	
	0	buffer_manager_access	
0x25	7:6	使用せず	
	5:0	buffer_manager_keyhole_address	
0x26	7:0	buffer_manager_keyhole_data	

表A.9.10 バッファマネージャアクセス及びキーホールレジスタ

【0646】

【表68】

アドレス (16進)	ビット No.	登録名	参照頁
0x00	7:0	使用せず	
0x01	7:2		
	1:0	cdb_base	
0x02	7:0		
0x03	7:0		
0x04	7:0	使用せず	
0x05	7:2		
	1:0	cdb_length	
0x06	7:0		
0x07	7:0		
0x08	7:0	使用せず	
0x09	7:0	cdb_read	
0x0A	7:0		
0x0B	7:0		
0x0C	7:0	使用せず	
0x0D	7:0	cdb_number	
0x0E	7:0		
0x0F	7:0		
0x10	7:0	使用せず	
0x11	7:0	tb_base	
0x12	7:0		
0x13	7:0		
0x14	7:0	使用せず	
0x15	7:0	tb_length	
0x16	7:0		
0x17	7:0		
0x18	7:0	使用せず	
0x19	7:0	tb_read	
0x1A	7:0		
0x1B	7:0		

表A.9.11 バッファマネージャの拡張アドレススペース(1/2)

【0647】

【表69】

アドレス (16進)	ビット No.	登録名	参照頁
0x1C	7:0	使用せず	
0x1D	7:0	tb_number	
0x1E	7:0		
0x1F	7:0		
0x20	7:0	使用せず	
0x21	7:0	buffer_limit	
0x22	7:0		
0x23	7:0		
0x24	7:4	使用せず	
	3	cdb_full	
	2	cdb_empty	
	1	tb_full	
	0	tb_empty	

表A.9.11 バッファマネージャの拡張アドレススペース(2/2)

【0648】

アドレス (16進)	ビット No.	登録名	参照頁
0x28	7	demux_access CED_H_CTRL[7]	
	6:4	buffer_error_code[2:0] CED_H_CTRL[6:4]	
	3:0	私的ハフマン制御ビット[3] は特殊CBPを送定する。[2] は 4/8ビット固定長CBPを送定する。	
0x29	7:0	parser_error_code CED_H_DMUX_ERR	
0x2A	7:4	使用せず	
	3:0	demux_keyhole_address	
0x2B	7:0	CED_H_KEYHOLE_ADDR	
0x2C	7:0	demux_keyhole_data CED_H_KEYHOLE	
0x2D	7	dummy_last_picture CED_H_ALU_REG0. r_dummy_last_frame_bit	
	6	field_into CED_H_ALU_REG0. r_field_into_bit	
	5:1	使用せず	
	0	continue CED_H_ALU_REG0. r_continue_bit	
0x2E	7:0	rom_revision CED_H_ALU_REG1	
0x2F	7:0	private register	
0x2F	7	CED_H_TRACE_EVENTを単一ステップに1を記入する。 1は、ステップが完了した時に読み取られる。	
	6	CED_H_TRACE_MASKを1にセットすると、単一ステップモードに入る。	
	5	CED_H_TRACE_RSTは、1.0 のシーケンスに配列された 場合には、部分的にリセットされる。	
	4:0	使用せず	

表A.9.12 ビデオデマルチプレクサレジスタ

【0649】

【表71】

アドレス (16進)	ビット No.	登録名	参照頁
0x00 0x0F	7:0	使用せず	
0x10 0x11	7:0	horiz_pelis r_horiz_pelis	
0x12 0x13	7:0	vert_pelis r_vert_pelis	
0x14	7:2	使用せず	
	1:0	buffer_size r_buffer_size	
0x15	7:0		
0x16	7:4	使用せず	
	3:0	pel_aspect r_pel_aspect	
0x17	7:2	使用せず	
	1:0	bit_rate r_bit_rate	
0x18 0x19	7:0		
0x1A	7:4	使用せず	
	3:0	pic_rate r_pic_rate	
0x1B	7:1	使用せず	
	0	constrained r_constrained	
0x1C	7:0	picture_type	
0x1D	7:0	h261_pic_type	

表A.9.13 ビデオデマルチプレクス拡張アドレススペース(1/9)

【0650】

【表72】

アドレス (16 進)	ビット No.	登 録 名	参照頁
0x1E	7:2	使用せず	
	1:0	broken_closed	
0x1F	7:5	使用せず	
	4:0	prediction_mode	
0x20	7:0	vbe_delay	
0x21	7:0		
0x22	7:0	私的レジスタ MPEG full_pel_fwd. JPEG pending_frame_change	
0x23	7:0	私的レジスタ MPEG full_pel_bwd. JPEG restart_index	
0x24	7:0	私的レジスタ horiz_sb_copy	
0x25	7:0	pic_number	
0x26	7:1	使用せず	
	1:0	sax_h	
0x27	7:1	使用せず	
	1:0	sax_v	
0x28	7:0	私的レジスタスクラッチ1	
0x29	7:0	私的レジスタスクラッチ2	
0x2A	7:0	私的レジスタスクラッチ3	
0x2B	7:0	N1 MPEG unused L.H. 261 ingob	
0x2C	7:0	私的レジスタ MPEG first_group. JPEG first_scan	
0x2D	7:0	私的レジスタ MPEG in_picture	
0x2E	7	dummy_last_picture_r_row_control	
	6	field_into	
	5:1	使用せず	
	0	継続する	
0x2F	7:0	row_revision	
0x30	7:2	使用せず	
	1:0	dc_huff_0	
0x31	7:2	使用せず	
	1:0	dc_huff_1	

表 A.9.13 ビデオマルチプレクス拡張アドレススペース(2/9)

【0651】

【表73】

アドレ ス (16進)	ビット No.	登 録 名	参照頁
0x32	7:2	使用せず	
	1:0	dc_huff_2	
0x33	7:2	使用せず	
	1:0	dc_huff_3	
0x34	7:2	使用せず	
	1:0	ac_huff_0	
0x35	7:2	使用せず	
	1:0	ac_huff_1	
0x36	7:2	使用せず	
	1:0	ac_huff_2	
0x37	7:2	使用せず	
	1:0	ac_huff_3	
0x38	7:2	使用せず	
	1:0	ta_0 r_ta_0	
0x39	7:2	使用せず	
	1:0	ta_1 r_ta_1	
0x3A	7:2	使用せず	
	1:0	ta_2 r_ta_2	
0x3B	7:2	使用せず	
	1:0	ta_3 r_ta_3	
0x3C	7:0	component_name_0 r_c_0	
0x3D	7:0	component_name_1 r_c_1	
0x3E	7:0	component_name_2 r_c_2	
0x3F	7:0	component_name_3 r_c_3	
0x40	7:0	私的レジスタ	
0x43			
0x40	7:0	r_dc_pred_0	
0x41	7:0		
0x42	7:0	r_dc_pred_1	
0x43	7:0		

表A.9.13 ビデオデマルチプレクス拡張アドレススペース(3/9)

[0 6 5 2]

[表 7 4]

アドレス (16進)	ビット No.	登録名	参照頁
0x44	7:0	r_dc_pred_2	
0x45	7:0		
0x46	7:0	r_dc_pred_3	
0x47	7:0		
0x48	7:0	使用せず	
0x4F			
0x50	7:0	r_prev_sht	
0x51	7:0		
0x52	7:0	r_prev_svt	
0x53	7:0		
0x54	7:0	r_prev_sbb	
0x55	7:0		
0x56	7:0	r_prev_svb	
0x57	7:0		
0x58	7:0	使用せず	
0x5F			
0x60	7:0	r_horiz_sbcnt	
0x61	7:0		
0x62	7:0	r_vert_sbcnt	
0x63	7:0		
0x64	7:0	horiz_macroblocks r_horiz_sbs	
0x65	7:0		
0x66	7:0	vert_macroblocks r_vert_sbs	
0x67	7:0		
0x68	7:0	private register r_restart_cnt	
0x69	7:0		
0x6A	7:0	restart_interval r_restart_int	
0x6B	7:0		
0x6C	7:0	private register r_blk_h_cnt	
0x6D	7:0	private register r_blk_v_cnt	

表A.9.13 ビデオデマルテプレクス拡張アドレススペース(4/9)

【0653】

【表75】

アドレス (16 進)	ビット No.	登 録 名	参照頁
0x6E	7:0	私的レジスタ r_cocpid	
0x6F	7:0	max_component_id r_max_cocpid	
0x70	7:0	coding_standard r_coding_std	
0x71	7:0	私的レジスタ r_pattern	
0x72	7:0	私的レジスタ r_fwd_r_size	
0x73	7:0	私的レジスタ r_bwd_r_size	
0x74	7:0	使用せず	
0x77			
0x78	7:2	使用せず	
	1:0	blocks_h_0 r_blk_h_0	
0x79	7:2	使用せず	
	1:0	blocks_h_1 r_blk_h_1	
0x7A	7:2	使用せず	
	1:0	blocks_h_2 r_blk_h_2	
0x7B	7:2	使用せず	
	1:0	blocks_h_3 r_blk_h_3	
0x7C	7:2	使用せず	
	1:0	blocks_v_0 r_blk_v_0	
0x7D	7:2	使用せず	
	1:0	blocks_v_1 r_blk_v_1	
0x7E	7:2	使用せず	
	1:0	blocks_v_2 r_blk_v_2	
0x7F	7:2	使用せず	
	1:0	blocks_v_3 r_blk_v_3	
0x7F	7:0	使用せず	
0xFF			
0x100	7:0	dc_bits_0[15:0] CED_H_KEY_DC_CPB0	
0x10F			
0x110	7:0	dc_bits_1[15:0] CED_H_KEY_DC_CPB1	
0x11F			

図 A. 9.13 ビデオデマルチプレクサ拡張アドレススペース(3/9)

【0654】

【表 7 6】

アドレス (16 進)	ビット No.	登 録 名	参照頁
0x120 0x13F	7:0	使用せず	
0x140 0x14F	7:0	ac_bits_0[15:0] CED_H_KEY_AC_CPB0	
0x150 0x15F	7:0	ac_bits_1[15:0] CED_H_KEY_AC_CPB1	
0x160 0x17F	7:0	使用せず	
0x180	7:0	dc_zssss_0 CED_H_KEY_ZSSSS_INDEX0	
0x181	7:0	dc_zssss_1 CED_H_KEY_ZSSSS_INDEX1	
0x182 0x187	7:0	使用せず	
0x188	7:0	ac_eob_0 CED_H_KEY_EOB_INDEX0	
0x189	7:0	ac_eob_1 CED_H_KEY_EOB_INDEX1	
0x18A 0x18B	7:0	使用せず	
0x18C	7:0	ac_zrl_0 CED_H_KEY_ZRL_INDEX0	
0x18D	7:0	ac_zrl_1 CED_H_KEY_ZRL_INDEX1	
0x18E 0x1FF	7:0	使用せず	
0x200 0x2AF	7:0	ac_huffval_0[161:0] CED_H_KEY_AC_ITOO 0	
0x2B0 0x2BF	7:0	dc_huffval_0[11:0] CED_H_KEY_DC_ITOO 0	
0x2C0 0x2FF	7:0	使用せず	
0x300 0x3AF	7:0	ac_huffval_1[161:0] CED_H_KEY_AC_ITOO 1	
0x3B0 0x3BF	7:0	dc_huffval_1[11:0] CED_H_KEY_DC_ITOO 1	

表A.8.13 ビデオデマルテプレクス並送アドレススペース(6/9)

【0655】

【表77】

アドレス (16 進)	ビット No.	登録名	参照頁
0x3C0 0x7FF	7:0	使用せず	
0x800 0xAC F	7:0	私的レジスタ	
0x800 0x80F	7:0	CED_KEY_TCOEFF_CPB	
0x810 0x81F	7:0	CED_KEY_CBP_CPB	
0x820 0x82F	7:0	CED_KEY_MBA_CPB	
0x830 0x83F	7:0	CED_KEY_MVD_CPB	
0x840 0x84F	7:0	CED_KEY_MTYPE_I_CPB	
0x850 0x85F	7:0	CED_KEY_MTYPE_P_CPB	
0x860 0x86F	7:0	CED_KEY_MTYPE_B_CPB	
0x870 0x88F	7:0	CED_KEY_MTYPE_H.261_CPB	
0x880 0x900	7:0	使用せず	
0x901	7:0	CED_KEY_HOSTROM_0	
0x902	7:0	CED_KEY_HOSTROM_1	
0x903 0x90F	7:0	CED_KEY_HOSTROM_2	
0x910 0xAB F	7:0	使用せず	

図 A. 9. 13 ビデオチャネルデプレックス拡張アドレススペース(7/9)

【0656】

【表78】

アドレス (16進)	ビット No.	登録名	参照頁
0xAC 0	7:0	CED_KEY_DMX_WORD_0	
0xAC 1	7:0	CED_KEY_DMX_WORD_1	
0xAC 2	7:0	CED_KEY_DMX_WORD_2	
0xAC 3	7:0	CED_KEY_DMX_WORD_3	
0xAC 4	7:0	CED_KEY_DMX_WORD_4	
0xAC 5	7:0	CED_KEY_DMX_WORD_5	
0xAC 6	7:0	CED_KEY_DMX_WORD_6	
0xAC 7	7:0	CED_KEY_DMX_WORD_7	
0xAC 8	7:0	CED_KEY_DMX_WORD_8	
0xAC 9	7:0	CED_KEY_DMX_WORD_9	
0xAC A 0xAC B	7:0	使用せず	
0xAC C	7:0	CED_KEY_DMX_AINCR	
0xAC D	7:0		

表A.9.13 ビデオマルチプレクス拡張アドレススペース(8/9)

【0657】

30 【表79】

アドレス (16進)	ビット No.	登録名	参照頁
0xAC E	7:0	CED_KEY_DMX_CC	
0xAC F	7:0		

表A.9.13 ビデオマルチプレクス拡張アドレススペース(9/9)

【0658】

【表80】

アドレス (16 進)	ビット No.	登 録 名	参照頁
	7:1	使用せず	
0x30	7:1	使用せず	
	0	iq_access	
0x31	7:2	使用せず	
	1:0	iq_coding_standard	
0x32	7:5	使用せず	
	4:0	test register iq_scale	
0x33	7:2	使用せず	
	1:0	test register iq_component	
0x34	7:2	使用せず	
	1:0	test register inverse_quantiser_prediction_mode	
0x35	7:0	test register jpeg_indirection	
0x36	7:2	使用せず	
	1:0	test register mpeg_indirection	
0x37	7:0	使用せず	
0x38	7:0	iq_table_keyhole_address	
0x39	7:0	iq_table_keyhole_data	

表A.9.14 逆量子化器レジスタ

【0659】

【表81】

アドレス (16 進)	登 録 名	参照頁
0x00:0x3F	JPEG逆量子化表0 MPEGデフォルトイントラ表	
0x40:0x7F	JPEG逆量子化表1 MPEGデフォルト非イントラ表	
0x80:0xBF	JPEG逆量子化表2 MPEG下方ロードイントラ表	
0xC0:0xFF	JPEG逆量子化表3 MPEG下方ロード非イントラ表	

表A.9.15 Iq表拡張アドレススペース

セクション A.10 「コード化データ入力」

本発明に基づくシステムは、処理するためにどの動画規格が入力されつつあるかを知らなければならない。今後、本システムは、先在しているトークン、又は、生バイトデータのいずれかを受け入れることができる。この生バイトデータは、次に、スタートコード検出器によってトークン内に配置可能である。

【0660】従って、コード化データ及びコンフィギュレーショントークンは、次に示す2つのルートを通じて空間デコードに供給可能である。

【0661】*コード化データ入力ポート

*マイクロプロセッサインターフェース (MPI)

使用するルートの選定は、アプリケーション及びシステム環境に依存する。例えば、データレートが低い場合には、デコーダチップ・セットを制御し、そして、システムビットストリームを多重化する両方の目的のために、1つの単一マイクロプロセッサを使用することが可能である。この場合、MPIを介してコード化データを入力をすることも可能である。代りに、コード化データレートが高い場合には、コード化データはコード化データホ

ートを介して供給されなければならない。

【0662】アプリケーションによっては、MPIとコード化データポート入力との混合体を使用することが適当である。

【0663】A. 10. 1 「コード化データポート」

【0664】

【表82】

信号名	入力/出力	記述
coded_clock	入力	入力回路の動作を制御する 30 MHz までに動作するクロック
coded_data[7:0]	入力	8ビットデータ値を転送するトークンポートを実現するために必要な標準11ワイヤ。このインタフェースの電氣的な説明についてはセクションA.4参照。オフチップ回路は、コード化されたデータをトークン内にバックしなければならない。
coded_extn	入力	
coded_valid	入力	
coded_accept	出力	
byte_mode	入力	この信号がhighである場合には、情報はトークンモードでなくバイトモードにおいて、コード化データポートを通して転送されなければならない。

表A.10.1 コード化データポート信号

本発明に基づくコード化データポートは、2つのモード、即ち、トークンモード及びバイトモードにおいて使用可能である。

【0665】A. 10. 1. 1 「トークンモード」

本発明において、バイトモードがローである場合には、コード化データポートは、正常な方法におけるトークンポートとして作動し、そして、コード化有効及びコード化アクセプトの制御の下にトークンを受け入れる。このインターフェースの電氣的動作の詳細についてはセクションA. 4を参照されたい。

【0666】信号バイトモードは、データ[7:0]、コード化extn、及びコード化有効と同時に、即ち、コード化clockの立ち上がりエッジにおいてサンプルされる。

【0667】A. 10. 1. 2 「バイトモード」

ただし、バイトモードがハイであれば、データの1つのバイトは、2線インターフェース制御信号コード化有効、及びコード化アクセプトの制御の下にデータ[7:0]において転送される。この場合、コード化extnは無視される。続いて、バイトは、入力モードが変更される時まで、オンチップにおいて、データトークンに組み立てられる。

【0668】1) トークンモードにおいて供給されたトークンの第1のワード(ヘッド)

2) 供給されたトークンの最後のワード(コード化extnはローになる)

3) バイトモードにおいて供給されたデータの第1のバ

イト。新しいデータトークンは、オンチップにおいて自動的に作成される。

【0669】A. 10. 2 「MPIを介したデータの供給」

トークンは、コード化データ入力レジスタにアクセスすることによってMPIを介して空間デコーダに供給可能である。

【0670】A. 10. 2. 1 「MPIを介したトークンの記入」

本発明のコード化データレジスタは、効率的なデータ転送を可能にするために、メモリマップ内において2バイトにグループ化される。8データビット、コード化データ[7:0]は1つの場所に配置され、また、制御レジスタ、コード化ビズイ、enable mpi入力、及びコード化extnは別の場所に配置される。表62、表63参照。

【0671】MPIを介してトークン入力用に構成された場合、1つの値がコード化データ[7:0]に記入される度に、現行トークンは、コード化extnの現行値を用いれ拡張される。あらゆるトークンの最後のワードがコード化データ[7:0]に記入される前に、コード化extnを0にセットすることに関してはソフトウェアに責任がある。

【0672】例えば、データトークンは、コード化extnに1を、又、コード化データ[7:0]に0x04を記入することによってスタートされる。次に、この新しいデータトークンのスタートは、処理のために、空間

デコーダへ供給する。

【0673】新しい8ビットの値がコード化データ

【7:0】に記入される度に、現行トークンが拡張される。例えば、別のトークンを導入するために、現行トークンを終わらせる場合に限り、コード化ext nは、再度アクセスされる必要がある。現行トークンの最後のワ

ードは、現行トークンの最後のワードのコード化データ【7:0】への記入によって後続されるコード化ext nへの0記入によって示される。

【0674】

【表83】

登録名		リセット状態	記 述
coded_extn	1 rw	x	トークンは、これらのレジスタに記入することにより、MPIを経て空間デコーダへ供給することができる。
coded_data[7:0]	8 w	x	
coded_busy	1 r	1	レジスタの状態は、空間レジスタが coded_data[7:0]に記入されたトークンを受入れ可能であるかどうかを示す。 値1は、インタフェースが使用中であって、データ受入れが不可能であることを示す。coded_busy=1である場合に、ユーザが coded_data[7:0] に記入を試みた場合における作動状態は未定である。
enable_apl_input	1 rw	0	この値は、コード化データポート(0)又はMPI(1)のどちらかを介して、空間デコーダへのコード化データ入力のレジスタによる制御を可能にする。

図A.10.2 コード化データ入力レジスタ

コード化データ【7:0】に先立ち、毎度、インタフェースがより多くのデータを受け入れる準備が整っているかどうかを判定するために、コード化ビジィは検査されなければならない。

【0675】A.10.3 「入力モード間のスイッチング」

適当な事前措置が講じられていることを条件として、データ入力モードを動的に変えることが可能である。一般に、任意の1つのルートを経たトークンの転送は、スイッチングモード前に完了していなければならない。

【0676】

【表84】

前モード	次モード	作 動 状 態
バイト	トークン	オンチップ回路は、作られつつあったDATAトークンの最終バイトとしてバイトモードにおいて供給された最終バイトを使用する(即ち <code>extn</code> 番ビットは0にセットされる)次のトークンを受け入れる前。
	MPI入力	
トークン	バイト	バイトモードを決定する以前に、当該トークンの完了に関しては(即ち0にセットされた情報の最終バイトの <code>extn</code> 番目のビットを用いて)トークンモードにおいてトークンを供給するオフチップ回路に責任がある。
	MPI入力	トークンモードにおいてトークンを供給するオフチップ回路が当該トークンを完了するまで(即ち0にセットされた情報の最終バイトの <code>extn</code> 番目のビットを用いる)、MPIを介した入力へのアクセスは許可されない(即ち <code>coded_busy</code> は1にセットされた状態に落ちる。)
MPI入力	バイト	制御ソフトウェアは、 <code>enable_mpi_input</code> が0にセットされる以前に、当該トークンを完了しなければならない(即ち0にセットされた情報の最終バイトの <code>extn</code> 番目のビットを用いる。)
	MPI入力	

表A.10.3 スイッチングデータ入力モード

バイトモードにおいて供給された第1のバイトによって、データトークンヘッダはオンチップにおいて生成可能である。バイトモードにおいて更に転送された全てのバイトは、今後、入力モードが変化するまでこのデータトークンに添付される。データトークンは必要だけ多くのビットを含むことができることを思い起こされたい。

【0677】MPIレジスタビット、コード化ビジィ、及び信号、コード化アクセプトは、当該空間デコーダがどのインターフェースにおいてデータを受け入れようとしているかを示す。これらの信号を正しく観察することにより、データが失われていないことが保証される。

【0678】A. 10. 4 「コード化データの受入れレート」

本発明において、入力回路は、トークンをスタートコード検出器に供給する(セクションA. 11参照)。スタートコード検出器は、データを、データトークンビットとして直列的に分析する。検出器の処理正常レートは、(コード化clock)のクロックサイクルにつき1ビットである。従って、デコーダは、コード化clockの8サイクル毎に、コード化データの1バイトを復号化する。ただし、例えば、非データトークンが供給される場合、或いは、コード化データ内にスタートコードが現れた場合のように、追加処理サイクルが必要とされることもある。この種のイベントが発生した場合、スタートコード検出器は、短時間に亘って、それ以上の情報の受け入れが不能とされる。

【0679】スタートコード検出器の後において、データは、第1ロジックコード化データバッファに供給する。このバッファがいっぱいになった場合、スタートコード検出器は、それ以上の情報を受け入れることができなくなる。

【0680】従って、それ以上のコード化データ(或いは、他のトークン)は、コード化データポートにおいて、又は、MPIを介して受け入れられる。この場合、スタートコード検出器は、それ以上の情報を受け入れることができない状態のままである。これは、信号コード化アクセプト及びレジスタコード化ビジィの状態によって示される。

【0681】コード化アクセプト、及び/又は、コード化ビジィを用いることにより、ユーザーにとって、コード化情報が一切失われないことが保証される。ただし、当該技術分野における通常の習熟者であれば、空間デコーダがデータ受け入れ不能である場合、システムは、新しく到着するコード化データを緩衝するか、(或いは、新しいデータの到着を停止する)ことを理解できるはずである。

【0682】A. 10. 5 「コード化データクロック」

本発明に基づき、空間デコーダにおけるコード化データポート、入力回路、及び他の機能は、コード化clockによって制御される。更に、このクロックは、メインデコーダクロックに対して非同期であっても差し支えない。データ転送は、オンチップにおけるデコーダクロック

クに同期する。

【0683】セクション A. 11 「スタートコード検出器」

A. 11. 1 「スタートコード」

当該技術分野において周知であるように、MPEG及びH. 261コード化動画ストリームは、スタートコードと呼ばれる識別可能なビットパターンを含む。JPEGにおいては、同様の機能は、マーカコードによって提供される。スタート/マーカコードは、コード化データストリームのシンタックスの有意部分を識別する。スタートコード検出器によって行われるスタート/マーカコードの分析は、コード化データのパーズングにおいては第1段階である。スタートコード検出器は、入力回路に後続する空間デコーダにおける第1ブロックである。

【0684】スタート/マーカコードパターンは、ビットストリーム全体を復号化することなしに識別可能であるように設計される。従って、スタート/マーカコードパターンは、本発明に従い、エラー回復及びデコーダスタートを助けるために使用できる。スタートコード検出器は、コード化データ構造内においてエラーを検出し、そして、デコーダのスタートを援助するための機能を提供する。

【0685】A. 11. 2 「スタートコード検出器レジスタ」

既に検討したように、多数のスタートコード検出器レジスタは、スタートコード検出器によって常時使用される。従って、データを処理するスタートコード検出器がデータを処理しつつある場合に、これらのレジスタにアクセスすることは、信頼度が低い。そのレジスタへのアクセス以前におけるスタートコード検出器の停止を保証することに関しては、ユーザの責任である。

【0686】レジスタスタートコード化etector accessは、スタートコード検出器を停止させ、ひいては、そのレジスタへのアクセスを可能にするために使用される。スタートコード検出器は、割込みを生成した後で、停止する。

【0687】スタートコード探索および全データモードの廃棄をいつ開始可能であるかと言うことに関しては、更に制約条件がある。これらの制約条件については、A. 11. 8、及びA. 11. 5. 1に記述済みである。

【0688】

【表85】

登 録 名	リセ ット 状 態	記 述
start_code_detector_access	1 0 iv	このレジスタへの記入1は、スタートコード検出器がそのレジスタへのアクセスを可能にして停止することを要求する。ユーザは、このレジスタから値1が読取られ動作が停止し、アクセスが可能になったことが示されるまで待機せねばならない

表A. 11. 1 スタートコード検出器レジスタ(1/5)

【0689】

【表86】

登録名	リセット	状態	記 述
illegal_length_count_event	1	0	JPEGデータ解析期間中に送込長さカウンティ
	rv		イベントが生じた場合には、長さカウンティ
illegal_length_count_mask	1	0	は、2未満の値を持つことが判明する。これは
	rv		JPEGデータ中のエラーの結果としてのみ発生
			する筈である。マスクレジスタを1にセットする
			と、割込みを生成可能であり、スタートコード検
			出器は停止する。このエラーが検出された場合に
			は（マスクレジスタが0にセットされている）エ
			ラーに続く作動状態は予測不可能である。
			A.11.4.1 参照
jpeg_overlapping_start_event	1	0	コード化規格がJPEG及びシーケンス 0xFF で
	rv		ある場合には、マークコードのロック中にこのイ
jpeg_overlapping_start_mask	1	0	イベントが生じる。このシーケンスは正当なスタッ
	rv		フィングシーケンスである。マスクレジスタが1
			にセットされている場合には、割込みが発生可能
			であり、そしてスタートコード検出器は停止する。
			A.11.4.2 参照
overlapping_start_event	1	0	コード化規格がMPEG又はH.261であり、スタ
	rv		ートコード探索中に重複スタートコードが発見さ
overlapping_start_mask	1	0	れた場合には、このイベントが生じる。マスクレ
	rv		ジスタが1にセットされている場合には、割込み
			が生成可能であり、スタートコード検出器が停止
			する。A.11.4.2 参照

表A.11.1 スタートコード検出器レジスタ(2/5)

【0690】

【表87】

登録名	リセット状態	記述
unrecognised_start_event	1 0 rv	未認識スタートコードに通過した場合にこのイベントが生じる。マスクレジスタが1にセットされると、新込みが生成可能であり、スタートコード検出器が停止する。ビットストリームから読取られたスタートコード値はレジスタの start_value において利用可能であり、一方においてはスタートコード検出器は停止される。A. 11.4.3 参照 正常動作期間中、start_value は最も最近検出されたスタートマークコードの値を含む。 H. 261 動作期間中は、start value の4つのLSBのみが用いられる。4つのMSBはゼロである。
unrecognised_start_mask	1 0 rv	
start_value	8 x ro	
stop_after_picture_event	1 0 rv	レジスタの stop_after_picture が1にセットされると、ピクチャの終端がスタートコード検出器を通過した後で画像停止イベントが生成される。マスクレジスタが1にセットされると、新込みが生成され、スタートコード検出器が停止される。 A. 11.5.1 参照 stop_after_pictureコードは、ピクチャの終端が検出された後で、0にリセットせず、従って直接クリアされる。
stop_after_picture_mask	1 0 rv	
stop_after_picture	1 0 rv	

表A.11.1 スタートコード検出器レジスタ(3/5)

【0691】

【表88】

登 録 名	リ セ ット 状 態	記 述
non_aligned_start_event	1 0 rv	ignore_non_alignedが1にセットされると、配列されたバイトでないスタートコードは無視される(正常データとして扱われる)。 ignore_non_alignedが0にセットされると、バイトの配列には無関係なH. 261及びMPEGスタートコードが検出される。 マスクレジスタが1にセットされると、イベントにより移送が発生し、スタートコード検出器が停止する。A. 11.6 参照 コード化規格がJPEGとして構成されている場合には、ignore_non_alignedは無視され、非配列スタートイベントは決して生成されない。
non_aligned_start_mask	1 0 rv	
ignore_non_aligned	1 0 rv	
discard_extension_data	1 1 rv	これらのレジスタが1にセットされると、空間デコードによって解読不可能な拡張又はユーザデータは、スタートコード検出器によって破棄される。A. 11.3.3 参照
discard_user_data	1 1 rv	
discard_all_data	1 0 rv	1にセットされると、全てのデータ及びトークンは、スタートコード検出器によって破棄される。これはFLUSHトークンが供給されるか、又はレジスタが直接0にセットされるまで継続する。このレジスタをリセットするFLUSHトークンは破棄され、スタートコード検出器により出力されない。A. 11.5.1 参照
insert_sequence_start	1 1 rv	A. 11.7 参照

表A. 11.1 スタートコード検出器レジスタ(4/5)

【0692】

【表89】

登 録 名	リ セ ツ ト 状 態	記 述
start_code_search	3 rw	このレジスタが0にセットされると、スタートコード検出器は正常に作動する。より高い値にセットされると、スタートコードの指定したタイプが検出されるまで、スタートコード検出器はデータを放棄する。指定されたスタートコードが検出されると、レジスタは0にセットされ、正常作動が継続する。A.11.3 参照
start_code_detector_coding_standard	2 rw	このレジスタは、スタートコード検出器により用いられるコード化規格を構成する。レジスタは直接的に、又はCODING_STANDARDトークンを用いてロード可能である。CODING_STANDARDトークンをスタートコード検出器が生成する場合には必ず (A.11.7.4参照)、このトークンは現行コード化規格構造を持つ。次に、このトークンは、デコーダチップセットの他の全ての部分により用いられるコード化規格を構成する。 A.21.1 及び A.11.7 参照
picture_number	4 rw	スタートコード検出器がデータストリーム (又は H.261又はJPEG等価) 内にピクチャスタートコードを検出した場合には、picture_numberの現行値を持つPICTURE_STARTトークンが生成される。

表 A. 11.1 スタートコード検出器レジスタ (5/5)

【0693】

【表90】

登 録 名	リ セ ツ ト 状 態	記 述
length count	16 rw	このレジスタは、JPEG長さカウンタの現行値を含む。このレジスタは、コード化データロックの制御の下で修正され、そしてスタートコード検出器が停止している場合に限り、MP1を介して読取られる。

表 A. 11.2 スタートコード検出器テストレジスタ

A. 11. 3 「スタートコードからトークンへの変換」
正常作動におけるスタートコード検出器の機能は、データストリーム内のスタートコードを識別し、そして、その次に、これらのコードを該当するスタートコードトークンに変換することである。最も簡単な場合、データは、1つの長い単一データトークンとしてスタートコード検出器に供給される。スタートコード検出器の出力は、スタートコードトークンを割り込み配置された多数の短かめのデータトークンである。

【0694】その代りに、本発明に基づき、スタートコード検出器への入力データは、多数の短かめのデータトークンに分割可能である。コード化データをデータトークンに分割する方法に関しては、nを整数とした場合、各データトークンは8 x nビットを含まなければならないということ以外には、制約は無い。

【0695】他のトークンは、スタートコード検出器の入力に直接供給可能である。この場合、トークンは、処理することなしに、スタートコード検出器を介して空間デコーダの他のステージへ供給される。これらのトーク

ンは、コード化データ内において、スタートコード位置の直前に限り挿入可能である。

【0696】A. 11. 3. 1 「スタートコードフォーマット」

本発明のスタートコード検出器によれば、3つの異なる

スタートコードフォーマットが認識される。これは、レジスタスタートコードディテクタコーディングスタンダードを介して構成される。

【0697】

【表91】

コード化規格	スタートコードパターン16進	スタートコード値のサイズ
MPEG	0x00 0x00 0x01 <value>	8 bit
JPEG	0xFF <value>	8 bit
H.261	0x00 0x01 <value>	4 bit

表A.11.3 スタートコードフォーマット

A. 11. 3. 2 「スタートコードトークン均等物」
スタートコードを検出した場合、スタートコード検出器は、当該スタートコードと関連した値を検討し、そして、適切なトークンを生成する。一般に、トークンは、関連MPEGシンタックスにちなんで命名される。ただし、当該技術分野における通常の習熟者であれば、ト

クンは、付加的ネーミングフォーマットに従うことを理解するはずである。現在選定されている符号化規格は、スタートコード値と生成されたトークンとの間の関係を構成する。この関係を表92に示す。

【0698】

【表92】

生成されるスタートコード トークン	スタートコード値			
	MPEG (16進)	H.261 (16進)	JPEG (16進)	JPEG (名称)
PICTURE_START	0x00	0x00	0xDA	SOS
SLICE_START ^a	0x01 to 0xAF	0x01 to 0x0C	0xD0 to 0xD7	RST ₀ to RST ₇
SEQUENCE_START	0xE3		0xD8	SOI
SEQUENCE_END	0xB7		0xD9	EOI
GROUP_START	0xB8		0xD0	SOF ₀ ^a
USER_DATA	0xB2		0xE0 to 0xEF	APP ₀ to APP ₇
			0xF0	COM
EXTENSION_DATA	0xB5		0xC0	JPG
			0xF0 to 0xF7	JPG ₀ to JPG ₇
			0x02 to 0xBF	RES
			0xC1 to 0xCB	SOF ₁ to SOF ₁₁
			0xCC	DAC
DHT_MARKER			0xC4	DHT
DNL_MARKER			0xDC	DNL
DQT_MARKER			0xDD	DQT
DR1_MARKER			0xD0	DR1

表A.11.4 スタートコード値からのトークン

a: このトークンは、スタートコード値により決定された値にロードされた8データビットフィールドを含む。

b: 基座系DCT符号化データのスタートを示す。

A. 11. 3. 3 「符号化規格の拡張機能」

符号化規格は、符号化規格によればその使用が現在確定していないデータストリーム内へのデータ埋め込みを可能にする多数のメカニズムを提供する。これは、特定メーカーに追加的な使い易さを提供する特定の「ユーザデータ」アプリケーションと見なすことも可能であり、そ

の代わりに、「拡張データ」と見なしても差し支えない。符号化規格オーソリティとしては、将来において符号化規格に機能を追加するために拡張データを使用する権利を留保した。

【0699】2つの明白なメカニズムが使用される。」

JPEGにおいては、ユーザ及び拡張データのブロックに

マーカコードを先行させる。ただし、H. 261においては、コード化データ内の追加情報ビットによって示される「追加情報」を挿入する。MPEGにおいては、これら双方の技術を使用できる。

【0700】本発明に従えば、スタート／マーカコードによって先行されるユーザ及び拡張データのMPEG/JPEGブロックは、スタートコード検出器によって検出可能である。H. 261/MPEGの「追加情報」は、本発明のハフマンデコーダによって検出される。

A. 14. 7「追加情報の受信」を参照されたい。

【0701】レジスタ、discardエクステンションデータ、及びdiscardユーザデータは、ユーザデータ及び拡張データを放棄するようにスタートコード検出器を構成することが可能である。このデータがスタートコード検出器で放棄されない場合には、このデータが動画デマックスに到達した時にアクセスすることが出来る。A. 14. 6、「ユーザ及び拡張データの受信」を参照されたい。

【0702】本発明の空間デコーダは、JPEGの基底線機能をサポートする。JPEGの非基底線機能は、空間デコーダによる拡張データと見なされる。従って、非基底線JPEGにとって、データに先行する全てのJPEGマーカコードは拡張データとして扱われる。

【0703】A. 11. 3. 4「JPEG表の定義」JPEGは、ダウンロードされたハフマン及び量子化器表をサポートする。JPEGデータにおいて、これらの表の定義は、マーカコードDNL及びDQTによって先行される。スタートコード検出器は、これらのマーカコードが検出された場合、トークンDHTMARKER及びDQTMARKERを生成する。これらのトークンは、動画デマックスに対して、後続するデータトークンがハフマン又は量子化器表について記述するコード化データを含むことを示す（JPEGにおいて記述されたフォーマットを用いる）。

【0704】A. 11. 4「エラーの検出」スタートコード検出器は、コード化データ内の特定エラーを検出可能であり、そして、エラーが検出された後においてデコーダによる回復を可能にする幾らかの機能を提供する（A. 11. 8、「スタートコードの探索」参照）。

【0705】A. 11. 4. 1「不当JPEG長さのカウント」

大抵のJPEGマーカコードは、それらと関連した16ビット長さのカウントフィールドを持つ。このフィールドは、どの程度の量がデータがこのマーカコードと関連しているかを示す。0及び1の長さカウントは不当である。不当長さは、データエラーに後続する場合に限り発生する。本発明において、イリーガルレンジスカウントマスクが1にセットされている場合、不当長さの発生は、割込みを生成する。

【0706】JPEGデータにおけるエラーからの回復は、JPEGにおいてスタートコードを検索することが困難であるため、付加的なアプリケーション特定データを必要とするものと推測される。段落A. 11. 8. 1参照）。

【0707】A. 11. 4. 2「スタート／マーカコードの重複」

本発明において、スタートコードの重複は、データエラーに後続する場合に限り発生する。MPEG配列バイト重複スタートコードを図73に示す。この場合、スタートコード検出器は、まず、ピクチャ・スタートコードのように見えるパターンを見る。次に、スタートコード検出器は、このピクチャ・スタートコードがグループスタートと重複していることを見る。従って、スタートコード検出器は、重複スタートイベントを生成する。更に、オーバーラッピングスタートマスクが1にセットされている場合には、スタートコード検出器は、割込み、及び停止を生成する。

【0708】2つのスタートコードのどちらが正しいか、及びデータエラーはどちらの原因であったかを判定することは不可能である。ただし、本発明に基づくスタートコード検出器は、最初のスタートコードを放棄し、そして、重複スタートコードイベントがサービスされた後で「第2のスタートコードが正しいものとして」第2のスタートコードの復号化を続行する。一連の重複スタートコードが存在する場合には、スタートコード検出器は、最後のコードを除くこれら全てのコードを破棄する（各重複スタートコードに関するイベントを生成する）。

【0709】非配列バイトシステムにおいては同様のエラーが可能である（H. 261、またはおそらくはMPEG）。この場合、イグノア・ノンアラインの状態も同様に考察されなければならない。図74は、見付かった最初のスタートコードは配列バイトであるが、しかし、非配列スタートコードと重複する例を示す。イグノア・ノンアラインが1にセットされている場合には、2番目の重複スタートコードは、スタートコード検出器によってデータとして扱われる。従って、重複スタートコードイベントは発生しない。これにより、発生しているかも知れないデータ通信エラーが隠される。イグノア・ノンアラインが0にセットされている場合には、スタートコード検出器は第2の非配列スタートコードを見る。そして、それが第1のスタートコードと重複していることを見る。

【0710】A. 11. 4. 3「非認識スタートコード」

スタートコード検出器は、認識されないスタートコードが検出された場合、割込みを生成することが出来る（アンレコグナイズドスタートマスク=1であれば）。この割込みの原因となったスタートコード値は、レジスタス

タートバリューから読み取り可能である。

【0711】スタートコード値0xB4（シーケンスエラー）は、チャンネルエラー又はメディアエラーを示すために、MPEGデコーダシステムにおいて使用される。例えば、訂正不可能なエラーが検出された場合には、このスタートコードは、ECC回路によってデータに挿入されることもあり得る。

【0712】A. 11. 4. 4 「イベント生成順序」本発明において、特定のコード化データパターン（エラー条件を示すことが多い）は、前述のエラー条件の中の1つ以上を短い期間中に起こさせる。従って、エラー条件に関してスタートコード検出器がコード化データを調査順序をつぎに示す。

【0713】1) 非配列スタートコード

2) 重複スタートコード

3) 未認識スタートコード

従って、非配列スタートコードがもう一方の後の方のスタートコードと重複する場合には、生成された最初のイベントは、非配列スタートコードと関連する。このイベントがサービスされた後において、スタートコード検出器の動作は継続し、短期間後に、重複スタートコードを検出する。

【0714】全ての非配列および重複スタートコードのテストが完了した後で、スタートコード検出器は単にスタートコードを認識することを試みる。

【0715】A. 11. 5 「デコーダのスタートおよび停止」

スタートコード検出器は、現行復号化タスクを完全に完了することを可能にし、そして、新しいタスクをスタート可能にする機能を提供する。

【0716】データセグメントはマーカコードをエミュレートする値を含むことができるので、JPEGコード化動画を用いるこれらの技術を使用することには制限がある（A. 11. 8. 1参照）。

【0717】A. 11. 5. 1 「復号化の明瞭な終了」

現行ピクチャに関するデータが完了してしまえば、割込みおよび停止を生成するようにスタートコード検出器を構成することができる。これは、ストップアフタピクチャ=1、及びストップアフタピクチャマスク=1をセットすることによって行われる。

【0718】ピクチャの終端がスタートコード検出器を一旦供給すると、フラッシュトークンが生成され（A. 11. 7. 2）、割込みが生成され、そして、スタートコード検出器が停止する。完成したばかりのピクチャは、通常の方法で復号化されることに注意されたい。アプリケーションによっては、デコーダチップ・セットの出力に到着したフラッシュを検出することが適切である場合もあり得る。この出力の到着は現行動画シーケンスの終了を示す。例えば、ディスプレイは、最後のピクチャ

出力に固着することもあり得る。

【0719】スタートコード検出器が停止すると、メディアとデコードチップとの間にユーザによって実現されたバッファに「トラップ」されている「古い」動画シーケンスからのデータもあり得る。レジスタdiscardallデータをセットすると、空間デコーダにこのデータを消費させ、そして、破棄させる。これは、フラッシュトークンがスタートコード検出器に到達するか、或いは、discardallデータがマイクロプロセッサインターフェースを介してリセットされる時まで、継続する。

【0720】「古い」シーケンスからのあらゆるデータが放棄されると、この段階において、新しいシーケンスに従って、デコーダが動作を開始する準備が整う。

【0721】A. 11. 5. 2 「全モードの破棄(discard)開始時期」

discardallデータレジスタに1が記入された直後に全モード破棄がスタートする。スタートコード検出器がデータを活発に処理している場合にこれが行われると、結果は予言不可能である。

【0722】スタートコード検出器イベント（非配列スタートイベント等）のうちのいずれかが割込みを生成した後で、全モード破棄は安全に開始されることが可能である。

【0723】A. 11. 5. 3 「新しいシーケンスの開始」

或るコード化データ内のどこで新しいコード化動画シーケンスがスタートするかが未知である場合には、スタートコード探索メカニズムを使用できる。このメカニズムは、シーケンスのスタートに先行するあらゆる不必要なデータを放棄する。A. 11. 8を参照されたい。

【0724】A. 11. 5. 4 「シーケンス間のジャンプ」

このセクションでは、前記の幾つかの技術のアプリケーションを示す。目的は、1つのコード化動画シーケンスの一部分からもう他の部分へ「ジャンプする」ことである。この例において、ファイリングシステムは、データの「ブロック」へのアクセスを可能にするに過ぎない。このブロック構造は、ディスク又はブロックエラー補正システムのセクタサイズから得ることも出来る。従って、コード化動画データにおける入り口及び出口の位置は、ファイリングシステムブロック構造と関係がない場合もあり得る。

【0725】ストップアフタピクチャ、及びdiscardallデータメカニズムは、古い動画シーケンスからの不必要なデータの破棄を可能にする。最後のファイリングシステムデータブロックの終了後、フラッシュトークンを挿入することにより、discardallデータモードはリセットされる。次に、スタートコード探索モードは、適当な入り口に先行する次のデータブロッ

ク内のあらゆるデータを放棄するために使用される。

【0726】A. 11. 6 「バイトの整列」

当該技術分野において周知であるように、異なる符号化計画は、データストリーム中のスタート/マーカコードのバイト整列に関して全く異なる見解を持つ。例えば、H. 261は通信を直列ビットとして見る。従って、スタートコードのバイト整列に関してはコンセプトがない。イグノア・ノンアライン=0をセットすることによって、スタートコード検出器は、任意のビット整列のスタートコードを検出することができる。ノンアラインスタートマスク=0をセットすることによって、スタートコード非配列割込みが抑圧される。

【0727】ただし、これとは対照的に、JPEGは、バイト整列が保証されているコンピュータ環境用として設計された。従って、マーカコードは、バイトが整列した場合に限り検出されるべきである。符号化規格がJPEGとして構成された場合には、イグノア・ノンアラインレジスタは無視され、そして、非整列スタートイベントは決して生成されない。ただし、将来の製品との互換性を保証するためにイグノア・ノンアライン=1、及び20 ノンアラインスタートマスク=0をセットすることが推奨される。

【0728】一方、MPEGは、通信（直列ビット）及びコンピュータ（バイト指向）システムの両方の必要性を満たすように設計された。MPEGデータ内のスタートコードは、通常、整列バイトでなければならない。た

だし、規格は、スタートコードに関して直列ビット探索が可能であるように設計されている（MPEGビットパターンは、どのようなビット整列であっても、スタートコードでない限り、スタートコードのようには見えない）。従って、MPEGデコーダは、直列データ通信におけるバイト整列の損失を許容するように設計可能である。

【0729】非整列スタートコードが見付かった場合には、通常、既に通信エラーが発生していたことを示す。エラーが直列ビット通信システムにおける「ビット・スリップ」である場合には、このエラーを含むデータは既にデコーダへ供給されている。このエラーは、デコーダ内において他のエラーを引き起こす可能性があるものと推測される。ただし、スタートコード検出器に到着する新しいデータは、この整列が失われた後で、続いて復号化可能である。

【0730】イグノア・ノンアライン=0、及びノンアラインスタートマスク=1をセットすることにより、非整列スタートコードが検出された場合には、割込みが生成可能である。責任の所在はアプリケーションに依存する。後続する全てのスタートコードは非整列である（バイト整列が復元されるまで）。従って、バイト整列が失われた後で、ノンアラインスタートマスク=0をセットすることが適切であることもあり得る。

【0731】

【表93】

	MPEG	JPEG	H.261
ignore_non_aligned	0	1	0
non_aligned_start_mask	1	0	0

表A.11.5 バイト配列の構成

A. 11. 7 「トークンの自動生成」

本発明において、スタートコード検出器によって出力されたトークンの大部分は、様々なピクチャ及び動画コード化規格の構文エレメントを直接反映する。これらの「自然」トークンに加えて、幾つかの有用な「発明された」トークンが生成される。これらの所有権を主張できるトークンの例はピクチャ・エンド、及びコーディング・スタンダードである。更に、符号化規格の間の或る程度40の構文的な差を除去し、そして、エラー条件の下で整頓するために、トークンが導入される。

【0732】この自動トークン生成は、コード化データの直列分析の後で行われる（図70「スタートコード検出器」参照）。従って、システムは、スタートコード検出器を介して空間デコーダの入力に直接供給されたトークン及びコード化データ内のスタートコードの検出に後続するスタートコード検出器によって生成されたトークンに平等に応答する。

【0733】A. 11. 7. 1 「ピクチャの終了表

示」

一般に、符号化規格は、ピクチャの終了を明白には送信しない。ただし、本発明のスタートコード検出器は、現行ピクチャが完了したことを示す情報を検出した場合、ピクチャ・エンドトークンを生成する。ピクチャ・エンドを生成させるトークンを次に示す。シーケンススタート、グループスタート、ピクチャ・スタート、シーケンスエンド、及びフラッシュ。

【0734】A. 11. 7. 2 「ピクチャ終了オプション後の停止」

レジスタストップアフタピクチャがセットされている場合には、スタートコード検出器は、ピクチャ・エンドトークンが供給した後で停止する。ただし、フラッシュトークンは、デコーダを介してコード化データの末端を「プッシュ」し、そして、システムをリセットするため、ピクチャ・エンドの後で挿入される。A. 11.

5. 1を参照されたい。

【0735】A. 11. 7. 3 「H. 261用シーケ

ンススタートの導入」

H. 261は、シーケンススタートに等価な構文エレメントを持たない表92参照)。インサートシーケンススタートレジスタがセットされている場合、スタートコード検出器は、その次のピクチャ・スタートの前に1つのシーケンススタートトークンがあることを保証する、即ち、スタートコード検出器がピクチャ・スタートの前にシーケンススタートを見ない場合には、一方のトークンが導入される。一方が既に存在する場合には、シーケンススタートは導入されない。

【0736】この機能は、MPEG又はJPEGと共に使用してはならない。

【0737】A. 11. 7. 4 「各シーケンスに対する符号化規格の設定」

スタートコード検出器を離れる全てのシーケンススタートトークンは、コーディング・スタンダードトークンによって常に先行される。このトークンには、スタートコード検出器の現行符号化規格がロードされる。これは、各新しい動画シーケンス用にセットされたデコーダチップ全体に対して符号化規格をセットする。

【0738】A. 11. 8 「スタートコードの探索」
本発明に基づくスタートコード検出器は、コード化データストリーム中の特定のタイプのスタートコードを探索するために使用できる。これは、デコーダが、いくらかのコード化データの構文内において、指定されたレベルからの復号化を再開することを可能にする（それに先行するあらゆるデータを放棄した後において）。このためのアプリケーションを次に示す。

【0739】*未知の位置において（例えば、ランダムアクセスの場合）コード化データファイルにジャンプした後におけるデコーダのスタート。

【0740】*データエラーの後において回復を援助するために、データ内の既知の点を探索する。

【0741】例えば、表94は、異なるコンフィギュレーションのスタートコードサーチに関して探索されたMPEGスタートコードを示す。等価H. 261及びJPEGスタート/マーカコードは、表92に見ることが出来る。

【0742】

20 【表94】

start_code_search	探索されるスタートコード...
0 ^a	正常作動
1	逆転（破棄カードとして作用する）
2	
3	シーケンススタート

start_code_search	探索されるスタートコード...
4	グループ又はシーケンススタート
5 ^b	ピクチャ、グループ又はシーケンススタート
6	スライス、ピクチャ、グループ又はシーケンススタート
7	次のスタート又はマーカコード

表A.11.6 スタートコード探索モード

a: FLUSHトークンは、スタートコード検出器をこの探索モードに置く。

b: これはリセット後のデフォルトモードである。

スタートコードサーチにゼロでない値が記入される場合、スタートコード検出器は、指定されたスタートコードが検出されるまで、入来する全てのデータを破棄することを開始する。次に、スタートコードサーチレジスタは0にリセットし、そして、正常作動が継続する。

【0743】スタートコードサーチは、ゼロ出ない値がスタートコードサーチレジスタに記入された後で、即座に開始する。スタートコード検出器がデータを活発に処理している時にこれが行われた場合には、結果は予測不可能である。従って、スタートコードサーチを開始する以前に、一切のデータが処理されていないようにスタートコード検出器が停止されていなければならない。任意のスタートコード検出器イベント（非整列スタートイベ

ント等）が割込みの生成を完了したばかりである場合には、スタートコード検出器は、常にこの条件にある。

【0744】A. 11. 8. 1 「JPEGと共にスタートコードサーチを使用する場合の制限」

殆どのJPEGマーカコードは、それらと関連した16ビット長カウンフィールドを持つ。このフィールドは、マーカコードと関連しているデータセグメントの長さを示す。このセグメントがマーカコードをエミュレートする値を含んでいても差し支えない。正常動作において、スタートコード検出器は、データのこれらのセグメント内のスタートコードを探さない。

【0745】或るJPEGコード化データへのランダムアクセスがこの種セグメントに「ランドする」する場合

には、スタートコード探索メカニズムを高信頼度を以て使用することは出来ない。一般に、J P E Gコード化動画は、ランダムアクセスのための入りを識別するために付加的な外部情報を必要とする。

【0746】セクション A. 12 「デコーダのスタート制御」

A. 12. 1 「デコーダスタートの概観」

デコーダにおいて、動画ディスプレイは、コード化データが最初に利用可能になった後で、通常、短い時間だけ遅れてディスプレイされる。この遅延の期間中、コード化データは、デコーダ内のバッファに集積する。バッファをこのように予充填することによって、復号化期間中にバッファが決して空にならないことを保証し、ひいては、デコーダが、規則的な間隔を以て、新しいピクチャを復号化することができることを保証する。

【0747】一般に、デコーダを正しくスタートするためには2つの機能が必要とされる。第1に、どれだけの量のデータがデコーダに供給されたかを測るメカニズムがなくてはならない。2番目に、新しい動画ストリームのディスプレイを防止するためのメカニズムがなければならぬ。本発明の空間デコーダは、どれだけの量のデータが到着したかを測定するために、その入力に近くピットカウンタを備え、そして、出力されつつある新しい動画ストリームがスタートすることを防止するために、その出力の近くに出力ゲートを備える。

【0748】これらの機能を制御するための複雑性に3つのレベルがある。

【0749】*出力ゲートは常に開いている

*基礎的制御

*上級制御

出力ゲートを常に開いた状態に保持することにより、コード化データがデコーダに到着し始めた後において、できる限り早期にピクチャ出力がスタートする。これは、静止画像を復号化する場合、或いは、ディスプレイが何か他のメカニズムによって遅延されつつある場合に適する。

【0750】基礎的制御と上級制御との間の差は、任意の時点において、デコーダのバッファ内に幾つの短い動画ストリームが収容可能かということである。大抵のA

アプリケーションにとっては基礎的制御で十分である。ただし、上級制御は、ユーザソフトウェアが、数個の非常に短い動画ストリームのスタートのデコーダによる管理を援助することを可能にする。

【0751】A. 12. 2 「MPEG動画バッファベリファイア」

MPEGは、一定データレートシステム用「動画バッファベリファイア (V B V)」について記述する。V B V情報を用いることにより、デコーダは、ピクチャをディスプレイを開始する以前に、そのバッファを予充填することが可能になる。再度説明すれば、この予充填により、デコーダのバッファが、復号化期間中に決して空にならないことが保証される。

【0752】要約すれば、各MPEGピクチャはv b v d e l a yパラメータを持つ。このパラメータは、第1のピクチャが復号化される以前に、「理想デコーダ」のコード化データバッファがコード化データでいっぱいになるために必要な時間を規定する。第1のピクチャのためのスタート遅延に注意すれば、後続する全てのピクチャの必要条件には自動的に適合する。

【0753】従って、MPEGは、スタートに関する必要条件を遅延として規定する。ただし、一定ビットレートシステムにおいては、この遅延は、ピットカウントに容易に変換できる。これは、本発明の空間デコーダのスタート制御動作の基礎である。

【0754】A. 12. 3 「ストリームの定義」

このアプリケーションにおいて、ストリームという用語は、MPEGにおけるシーケンスという用語との混乱を避けるために使われる。従って、ストリームは、アプリケーションにとって関心のある動画データの量を意味する。従って、1つのストリームが多数のMPEGシーケンスであるか、或いは、1つのピクチャでありうる。

【0755】この章に記述されるデコーダスタート機能は、1つのストリーム内の第1ピクチャに関するV B V必要条件に適合することに関係する。当該ストリーム内の後続ピクチャの必要条件には自動的に適合する。

【0756】A. 12. 4 「スタート制御レジスタ」

【0757】

【表95】

注 記	ビット数	注 記
startbs_access CED_BS_ACCESS	1 rw	0 このレジスタに1を記入すると、ビットカウンタ及びゲートクロックステップに、それら両レジスタへのアクセスを可能にする。
bit_count CED_BS_COUNT	8 rw	0 このビットカウンタは、コード化データスタートコード検出時に1にインクリメントされる。
bit_count_prescale CED_BS_PRESCALE	3 rw	0 bit_count は一度インクリメントするために必要なビット数を2 ^{bit_count} である。ビットカウンタは、FLUSHトークンがビットカウンタをバスした状態でカウントを開始する。ビットカウンタはゼロにリセット、次にビットカウンタ値に達した後にインクリメントを停止する。
bit_count_target CED_BS_TARGET	8 rw	x このレジスタは、ビットカウンタ値を指定する。既知のイベントに、次の条件が満たされる場合には必ず応答される。即ち bit_count >= bit_count_target
target_met_event BS_TARGET_MET_EVENT	1 rw	0 ビットカウンタ値に達した場合には、このイベントが生成される。マスキングレジスタ1にセットされると、応答が生成可能である。但し、ビットカウンタは、データ送達を停止しない。ビットカウンタがその値でインクリメントした場合、このイベントが生成される。ビットカウンタの値より少ないか又は等しい値が記入された場合にもこのイベントが生成される。bit_count_targetに0を記入すると必ず既知のイベントを生成する。
target_met_mask	1 rw	0

図A.12.1 デコーダスタートアップレジスタ

【0758】

【表96】

注 記 名	レジスタ番号	ビット	説 明
counter_flushed_event BS_FLUSH_EVENT	1	0	FLUSHトークンがビットカウンタ0番を通過する前に、このイベントが起る。マスクレジスタが1にセットされている場合には、読み込み可能であり、ビットカウンタは停止する。
counter_flushed_too_early_event BS_FLUSH_BEFORE_TARGET_VEL_EVENT	1	0	FLUSHトークンがビットカウンタ0番を通過し、ビットカウンタが読み込まれている場合に、このイベントが起る。マスクレジスタが1にセットされている場合には、読み込み可能であり、ビットカウンタは停止する。
flush_queue CED_BS_QUEUE	1	0	このレジスタを1にセットすると、マイクロプロセッサをフットする必要があるゲート用ロジックを有効にする。このレジスタが0にセットされている場合には、出力ゲート制御ロジックは、出力ゲートの動作を自動的に制御する。
enable_stream CED_BS_ENABLE_NXT_STM	1	0	オフチップキューが利用される場合には、enable_streamへの記入により、ストリームの開始が通知された後で、出力ゲートの動作状態を制御する。このレジスタに1を記入すると、出力ゲートを動作可能にする。accept_enable 読み込みが完了される前に、レジスタはリセットされる。
accept_enable_event BS_STREAM_END_EVENT	1	0	このイベントは、FLUSHトークンが出力ゲートを通過し（ゲートを開き）て、そしてゲートが動作可能にするためにインエーブルが利用できたことを示す。
accept_enable_mask	1	0	マスクレジスタが1にセットされると、読み込み可能となり、enable_stream レジスタがリセットになる。

図A.12.1 デコーダスタートアップレジスタ

A. 12. 5 「常時開出力ゲート」

出力ゲートは、開いた状態を維持するように構成可能である。静止画像が復号化されつつある場合、或いは、動画デコーダのスタートを管理するために他の何等かのメカニズムが利用可能である場合には、このコンフィギュレーション構成が適切である。

【0759】リセット後に必要とされるコンフィギュレーションを次に示す（スタートアップアクセスに1を記入することによってスタート制御ロジックへのアクセスが得られる）。

【0760】* オフチップキュー=1をセットする
* イネーブルストリーム=1をセットする
* 全てのデコーダスタートイベントマスクレジスタが0にセットされ、それらのレジスタの割込みを無能にすることを保証する（これは、リセット後のデフォルト状態である）。（これが出力ゲートを開いた状態に保持する理由についての説明に関してはA. 12. 7. 1を参照されたい）。

【0761】A. 12. 6 「基本動作」

本発明において、殆どのMPEG動画アプリケーションにとっては、スタートロジックの基礎的な制御で充分である。このモードにおいて、ビットカウンタは出力ゲ

と直接通信する。フラッシュトークンによって指示されるように、動画ストリームのエンドが出力ゲートを供給する時、出力ゲートは自動的に閉じる。ストリームがそのスタートビットカウンタを達成した時にビットカウンタ回路により1つのイネーブルが供給されるまで、当該ゲートは閉じた状態を維持する。

【0762】リセットの後で必要とされるコンフィギュレーションを次に示す（スタートアップアクセスに1を記入することによりスタート制御ロジックへのアクセスが得られる）。

【0763】*コード化データのほぼ予測範囲に対してビットカウントプリスケールをセットする。

【0764】*このエラー条件を検出可能にするためにカウンタflushed too earlyマスク=1をセットする。

【0765】次に示す2つの割込みサービスルーチンが必要とされる。

【0766】*新しい各ストリーム内の第1ピクチャ用v b v遅延値を得るための動画デマルチプレクスサービス。

【0767】*この条件に反応するためのカウンタflushed too earlyサービス。

【0768】新しい動画ストリーム用v b v遅延を復号化する時(即ち、フラッシュ後において第1ピクチャが動画デマルチプレクサに到着する時)、この動画デマルチプレクサ(別名動画パーザ)は割込みを生成することが出来る。割込みサービスルーチンは、ビットカウンタターゲット用の適切な値を計算し、そして、それを記入しなければならない。ビットカウンタがこの目標に到達した時、このビットカウンタは、ビットカウンタと出力ゲートとの間の短いキューに1つのイネイブルを挿入する。出力ゲートが開くと、このゲートはこのキューから1つのイネイブルを除去する。

【0769】A. 12. 6. 1 「別のストリーム終了直後における新ストリームの開始」

一例として、終わろうとしていMPEGストリームをAと呼び、そして、開始しようとしているMPEGストリームをBと呼ぶこととする。Aのエンドの後でフラッシュトークンが挿入されなければならない。このトークンは、そのコード化データの最後のデータをデコードを通して押し、デコーダの様々なセクションに新しいストリームが予測されることを警告する。

【0770】通常、ビットカウンタはゼロにリセット済みであり、Aは既にそのスタート条件に適合した状態である。フラッシュの後で、ビットカウンタは、ストリームB内のビットをカウントし始める。動画デマックスが、ストリームB内の第1ピクチャからv b v遅延を復号化し終わると、割込みが生成されて、ビットカウンタを構成可能にする。

【0771】ストリームAのエンドをマークするフラッシュが出力ゲートを通過するにつれて、ゲートが閉じる。ゲートは、Bがそのスタート条件に適合するまで、閉じた状態を維持する。例えば、ストリームB及びバッファの深さのためのスタート遅延のような多数の要因に応じて、出力ゲートが閉じた時、Bが既にそのスタート条件条件に適合済みであることが可能である。この場合、1つのイネイブルはキュー内において待機しており、そして、出力ゲートは即座に開く。そうでない場合には、ストリームBは、そのスタート必要条件に適合するまで、待たなければならない。

【0772】A. 12. 6. 2 「短いストリームの連続」

ビットカウンタと出力ゲートとの間に位置するキューの容量は、3つの個別動画ストリームをそれらのスタート条件に適合可能にし、そして、復号化を終了させるために前のストリームを待つことを可能にするために十分である。本発明において、非常に短いストリームが復号化されつつある場合、或いは、オフチップバッファが復号化中のピクチャフォーマットと比較して非常に大きい場合に限りこの条件が発生する。

【0773】図78において、ストリームAは復号化されつつあり、そして、出力ゲートは開いている。スト

ームB及びCは、それらのスタート条件に適合し、そして、空間デコーダによって管理されたバッファ内に完全に含まれる。ストリームDは、空間デコーダの入力に依然として到着しつつある。

【0774】ストリームB及びC用のエネイブルは、キュー内に在る。従って、ストリームAが完了する、Bは即座にスタート可能である。同様に、CはBの直後に後続可能である。

【0775】Dがそのスタート目標に適合するばあいにAが出力ゲートを依然として通過中である場合には、1つのイネイブルがキューに加えられ、キューを満たす。Dのエンドがビットカウンタを通過する時まで、一切のイネイブルがキューから除去されなかった場合には(すなわち、Aは、出力ゲートを通過中であるし)、新しいストリームは、ビットカウンタを経てスタートすることが出来ない。Bが出力ゲートを供給できるように出力ゲートは開いた状態にあるので、Aが完了し、そして、1つのイネイブルがキューから除去されるまで、コード化データは、入力において、ホールドアップされる。

【0776】A. 12. 7 「上級動作」

本発明に基づき、スタートロジックの上級制御は、A. 12. 6 「基礎動作」において記述されているようにユーザーソフトウェアが作動可能化キューの長さを制限なしに拡張することを可能にする。動画デコーダが、A. 12. 6. 2 「短いストリームの連続」において記述されているよりも長い一連の短い動画ストリームを収容しなければならない場合に限り、このレベルの制御が必要とされる。

【0777】システムの基礎動作に必要とされるコンフィギュレーション構成に加えて、リセットの後で、次に示すコンフィギュレーションが必要とされる(スタートアップアクセスに1を記入することによってスタート制御ロジックへのアクセスが得られる)。

【0778】*オフチップキュー=1をセットする。

【0779】*キューから1つのイネイブルが除去された場合、割込みを作動可能化するためにアクセプトイネイブルマスク=1をセットする。

【0780】*ストリームのビットカウンタ標的が達成された場合、割込みを作動可能化するためにターゲットmetマスク=1をセットする。

【0781】次に示す2つの付加的割込みサービスルーチンが必要とされる。

【0782】*受け入れ可能化割込み

*標的達成割込み

標的達成割込みターゲットが発生した場合、サービスルーチンは、そのオフチップ作動可能化キューに1つのイネイブル加えなければならない。

【0783】A. 12. 7. 1 「出力ゲートロジックの作動態様」

イネーブルストリームレジスタに1を記入すると、1つのイネーブルが短いキューにロードされる。

【0784】フラッシュ（ストリームのエンドをマークする）が出力ゲートを供給すると、ゲートは閉じる。利用可能な1つのイネーブルがキューのエンドに在る場合には、ゲートが開いて、そして、アクセプトイネーブルイベントを生成する。アクセプトイネーブルマスクが1にセットされる場合には、割込みが生成可能であり、そして、キューのエンドから1つのイネーブルが除去される（レジスタイネーブルストリームはリセットされる）。

【0785】ただし、アクセプトイネーブルマスクがゼロにセットされる場合には、アクセプトイネーブルイベントに続いて割込みは生成されず、そして、当該イネーブルはキューのエンドから除去されない。このメカニズムは、A. 12. 5に述べられているように、出力ゲートを開いた状態に保つために使用できる。

【0786】A. 12. 8 「ビットカウント」
ビットカウンタは、フラッシュトークンがこのカウンタを通過した後で、カウントを開始する。このフラッシュトークンは、現行動画ストリームのエンドを示す。この点に関して、ビットカウンタは、ビットカウントtargetレジスタにセットされているビットカウント目標に適合するまでカウントを継続する。次に、目標適合イベントが生成され、そして、ビットカウンタがゼロにリセットし、そして、次のフラッシュトークンを待つ。

【0787】更に、ビットカウンタは、最大カウント（255）に到達すると、インクリメントを停止する。

【0788】A. 12. 9 「ビットカウントプリスケール」

本発明において、ビットカウンタを1度インクリメントするには2の(bit count prescale+1)乗×512ビットが必要である。更に、ビットカウントプリスケールは、0と7との間に値を保持できる3ビットレジスタである。

【0789】

【表97】

n	レンジ (ビット)	解像度 (ビット)
0	0 to 252144	1024
1	0 to 524288	2048
2	0 to 31457280	122880

表A. 12. 2 例ビットカウンタレンジ

動画ストリームの幾つかの要素は既にトークン化されているので（例えば、スタートコード）、ビットカウンタは近似的であり、従って、非データトークンを含む。

【0790】A. 12. 10 「早過ぎるカウンタフラッシュ」

ビットカウント目標の達成以前に、フラッシュトークンがビットカウンタに到着する場合には、割込みの原因となり得るイベントが生成される（カウンタflushed too earlyマスク=1である場合）。割込みが生成されれば、ビットカウンタ回路が停止し、それ以上のデータ入力を阻止する。このイベントが発生した後において何時出力ゲートを開くかを決定するのはユーザソフトウェアの責任である。ビットカウント目標として0を記入することによって開くように出力ゲートを作成することが出来る。2、3のピクチャだけ継続する動画ストリームの復号化を試行する場合に限り、これらの状況が発生するはずである。

【0791】セクション A. 13 「バッファ管理」
空間デコーダは、2つの論理的データバッファ、即ち、コード化データバッファ（CDB）及びトークンバッファ（TB）を管理する。

【0792】CDBは、スタートコード検出器とハフマンデコーダの入力との間でコード化データを緩衝する。これは、低データレートコード化動画データに対して緩衝作用を提供する。TBは、ハフマンデコーダの出力と空間動画復号化回路（逆モデラ、量子化器、及びDCT）の入力との間のデータを緩衝する。この2番目の論理的バッファは、変動するデータ量を持つピクチャの処理を収容するために処理時間が拡張を含むことを可能にする。

【0793】双方のバッファは、物理的には1つの単一オフチップDRAMアレイに保持される。これらのバッファ用アドレスは、バッファマネージャによって生成される。

30 【0794】A. 13. 1 「バッファマネージャレジスタ」

空間デコーダバッファマネージャは、デバイスがリセットされた直後に、1度だけ構成されるように意図されている。正常作動中は、バッファマネージャの再構成に関する必要条件はない。

【0795】空間デコーダからリセットが除去された後で、バッファマネージャは停止させられ（バッファマネージャアクセスレジスタを1にセットする）、コンフィギュレーション化を待つ。レジスタが構成された後で、40 バッファマネージャアクセスは0にセット可能であり、復号化を開始することができる。

【0796】バッファマネージャ作動中はバッファマネージャに使用されているレジスタは、殆どの場合、高信頼度を以てアクセスすることは不可能である。あらゆるバッファマネージャレジスタにアクセスしようとする場合には、それ以前に、バッファマネージャアクセスを1にセットしなければならない。これは、バッファマネージャアクセスから値1が読み取られるまで、待ちプロトコルに従うことを必須にする。バッファの条件を監視するために、例えばcddbfull、及びcddbempty

のようなレジスタをポーリングする場合、アクセスの確保および放棄に要する時間について考慮しなければならない。

【0797】
【表98】

注 意 名	ビット数	注 意
buffer_manager_status	1 rw	このアクセスビットは、バッファマネージャの動作を停止し、その結果、各レジスタへの高速転送アクセスが不可能である。A. 6.4.1参照。注：このアクセスレジスタは、リセット後のデフォルト状態が1であり、即ちリセット後にバッファマネージャを停止させ、マイクロプロセッサのフェーズを待機状態とするので一般的でない。

注 意 名	ビット数	注 意
buffer_manager_keyhole_address	6 rw	次に示すバッファマネージャレジスタに使用される拡張アドレスベースへのキーホールアクセス。キーホールを介してのアクセスレジスタに属する各時間についてはA. 6.4.3参照のこと。
buffer_manager_keyhole_data	8 rw	
buffer_limit	18 rw	これは空間デコーダに提供された...アレイの全サイズを指定する。すべてのバッファアドレスは...このバッファサイズであり、従って宛先されたDRAM内に組み込まれる。
cdb_base	18 rw	これらのレジスタは、コード化データ(cdb)及びトークン(tb)バッファのベースを指定する。
tb_base		
cdb_length	18 rw	これらのレジスタは、コード化データ(cdb)及びトークン(tb)バッファの長さ(即ち、サイズ)を指定する。
tb_length		
cdb_read	18 ro	これらのレジスタは、バッファベースからのオフセットを保持し、現在のデータを次に読むべきものを指示する。
tb_read		
cdb_number	18 ro	これらのレジスタは、現在バッファ内に保持されているデータの量を示す。
tb_number		
cdb_full	1 ro	コード化データ(cdb)及びトークン(tb)バッファが満杯のときには、これらのレジスタは1にセットされる。
tb_full		
cdb_empty	1 ro	コード化データ(cdb)及びトークン(tb)バッファが空のときには、これらのレジスタは1にセットされる。
tb_empty		

図A. 13.1 バッファマネージャレジスタ

A. 13. 1. 1 「バッファマネージャのポインタの値」

一般に、データは、空間デコーダとオフチップDRAMとの間において64バイトのバーストとして転送される(DRAMの高速ページモードを用いる)。全てのバッファポインタ及び長さレジスタは、これらの64バイト(512ビット)データのブロックを意味する。従って、バッファマネージャの18ビットレジスタは、256kブロック線形アドレススペース(即ち、128Mb)を記述する。

【0798】64バイト転送は、DRAMインターフェースの幅(8、16、または32ビット)からは独立している。

【0799】A. 13. 2 「バッファマネージャレジスタの使用」

空間デコーダバッファマネージャは、2つの類似したバッファを定義する2組のレジスタを持つ。バッファリミットレジスタは、メモリスペースの物理的な上限を定義する。全てのアドレスは、この数を法として計算される。

【0800】利用可能なメモリの限界内において、各バッファの範囲は、2つのレジスタ、即ち、バッファベ

ース(cdbbase、及びtbbase)、及びバッファ長さ(cdblength、及びtblength)により定義される。バッファが使用可能になるためには、ここまでに記述した全てのレジスタがそれ以前に構成されていなければならない。

【0801】各バッファの現在の状態は、4つのレジスタにおいて視ることができる。バッファリードレジスタ(cdbread、及びtbread)は、そこから次にデータが読み出されるバッファベースからのオフセットを示す。バッファナンバーレジスタ(cdbnumber、及びtbnumber)は、バッファによって現在保持されているデータの量を示す。状態ビットcdbfull、tbfull、cdbempty、及びtbemptyは、バッファが満杯か、或いは、空であるかを示す。

【0802】A. 13. 1. 1に述べられているように、前述の全てのレジスタに関する単位は512ビットのデータブロックである。従って、コード化データバッファ内のビット数を求めるためには、cdbnumberから読み出された値を512を乗じなければならない。

【0803】A. 13. 3 「ゼロバッファ」

「リアルタイム」必要条件が適用されない静止画像用アプリケーション（例えば、JPEGを使用）は、バッファマネージャによってサポートされた大きいオフチップバッファを必要としない。この場合、コード化データバッファ及びトークンバッファに128ビットストリームオンチップFIFOを供給するためにバッファマネージャを無視するように（レジスタゼロバッファに1を記入することによって）DRAMインターフェースを構成可能である。

【0804】更に、ゼロのバッファオプションは、低データレートにおける小さいピクチャフォーマットによる動作を操作するアプリケーションに適する場合もあり得る。

【0805】注記：レジスタゼロバッファは、DRAMインターフェースの一部であるので、DRAMインターフェースのポストリセット構成化の期間中に限ってセットされなければならない。

【0806】A. 13. 4 「バッファオペレーション」

バッファを介して行われるデータ転送は、ハンドシェイクプロトコルによって制御される。従って、バッファが満杯であるか、又は、空である場合にはデータエラーが発生しないことが保証される。バッファが満杯である場合には、データをバッファに送ろうとする回路は、バッファにスペースができるまで、停止させられる。バッファが満杯状態を継続する場合には、当該バッファの上流に配置された更に処理し続けるステージは、空間デコーダがその入力ポートにおいてデータを受け入れることが

出来なくなるまで、停止する。同様に、バッファが空である場合には、バッファからデータを除去しようとする回路は、データが利用可能になる時まで、停止する。

【0807】A. 13. 2に示すように、コード化データ及びトークンバッファの位置およびサイズは、バッファベース及び長さレジスタによって指定される。これらのレジスタを構成すること、及び2つのバッファの間のメモリー用法に矛盾がないことを保証することはユーザの責任である。

【0808】セクションA. 14 「動画デマルチプレクサ」

動画パーザとも呼ばれる動画デマルチプレクサは、コード化データをスタートコード検出器によってスタートされたトークンに変換するというタスクを完了する。動画デマルチプレクサには4つの主要処理ブロックがある、即ち、パーザステートマシン、ハフマンデコーダ（ITODを含む）、マクロブロックカウンタ、及びALUである。

【0809】パーザ又はステートマシンは、コード化動画データの構文に従い、そして、他のユニットを指示する。ハフマンデコーダは、可変長コード化（VLC）データを整数に変換する。マクロブロックカウンタは、現在復号化されつつあるピクチャセクションのトラックを保持する。ALUは、必要な算術計算を行う。

【0810】A. 14. 1 「動画デマルチプレクサレジスタ」

【0811】

【表99】

注 記	レジスタ番号	注 記
denus_access CED_H_CTRL[7]	1 rw	このアクセスビットは、ビデオマックスの各レジスタへのアクセス権限を再取得のために、ビデオマックスの動作を停止する。 A. 8. 4. 1 参照
huffman_error_code CED_H_CTRL[8:4]	3 ro	ビデオマックスが、huffman_event 通知リクエスト生成の処理を停止すると、この3ビットレジスタは、通知が生成された理由を示す値を保持する。 A. 14. 5. 1 参照
parser_error_code CED_H_DMUX_ERR	8 ro	ビデオマックスが、parser_event 通知リクエスト生成の処理を停止すると、この8ビットレジスタは、通知が生成された理由を示す値を保持する。 A. 14. 5. 2 参照
denus_keyhole_address CED_H_KEYHOLE_ADDR	12 rw	ビデオマックスの拡張アドレスへのキーホールアクセスは完了する。キーホールを介したレジスタへのアクセスに関する詳細については、A. 8. 4. 3 参照
denus_keyhole_data CED_H_KEYHOLE	8 rw	図A. 14. 2、A. 14. 3 及びA. 14. 4 は、キーホールを介してアクセス可能なレジスタについて示す。

図A. 14. 1 トップレベルビデオマックスレジスタ

【0812】

【表100】

レジスタ名	ビット数	注 記
dummy_last_picture CED_H_ALU_REG0 r_res_control r_dummy_last_frame_bit	1 rw	0 このレジスタにセットされると、ビデオマックスはMPEGシーケンスの最終ピクチャとしてのデマントラピクチャに属する状態を生成する。この状態は、最終P又は最終デコーダからの1ピクチャをフラッシュするために最終ピクチャ再帰を待機 (A.14.3.5 [ピクチャシーケンス再帰待機時間]) 後に再帰デコーダを再帰する際に有効である。次に示す場合には、デマントラピクチャは不要である。 * 再帰デコーダ再帰待機時に生成されていない場合。 * もう一方のMPEGシーケンスが即座に生成される場合 (これは最終ピクチャのフラッシュを待機)。 * コード化規格がMPEGでない場合。
field_info CED_H_ALU_REG0 r_res_control r_field_info_bit	1 rw	0 このレジスタにセットされると、任意のMPEG extra_information_pictureの最初のバイトがFIELD_INFORMATION内に記憶される。 A.14.7.1 参照
continue CED_H_ALU_REG0 r_res_control r_continue_val	1 rw	0 このレジスタは、デコーダによって生成された場合、レジスタが入れようとされるエクストラ、ユーザ又は拡張データの長をユーザソフトウェアにより制御可能にする。 A.14.5 及び A.14.7 参照
res_revision CED_H_ALU_REG1 r_res_revision	8 rw	0 リセットの直ぐ後に置いて、これは、マイクロコードRCM識別番号のコピーを保持する。このレジスタは、コード化デコーダ生成したデータを制御ソフトウェアに転送するために用いられる。 A.14.8 [ユーザ及び拡張データの受取り] 及び A.14.1 [エクストラ情報データの受取り] 参照

図A.14.1 トップレベルビデオマックスレジスタ

【0813】

【表101】

レジスタ名	ビット数	注 記
huffman_event	1 rw	0 ハフマンイベントは、最終データ内にエラーが検出された場合に生成される。これらのイベントに関してはA.14.5.1 参照。マスクレジスタ1にセットされている場合は、読み込みの生成が可能であり、ビデオマックスは停止する。マスクレジスタ0にセットされている場合は、読み込みは生成されず、ビデオマックスはエラーからの回復を待機する。
huffman_mask	1 rw	0
parser_event	1 rw	0 パーザイベントは、コード化データ内のエラー、又はソフトウェアの介入を必要とするビデオマックスにおける状態遷移に発生可能である。これらのイベントに関してはA.14.5.1 参照。マスクレジスタ1にセットされている場合は、読み込みが生成可能であり、ビデオマックスは停止する。マスクレジスタ0にセットされている場合は、読み込みは生成されず、ビデオマックスは回復を待機する。
parser_mask	1 rw	0

図A.14.1 トップレベルビデオマックスレジスタ

【0814】

40 【表102】

レジスタ名	ビット数	注
component_name_0 component_name_1 component_name_2 component_name_3	8 rv	x JPEG符号化時、レジスタ component_name_n は、どの色成分が成分ID n となるかを（アプリケーションに）示す8ビット値を保持する。
horiz_delta vert_delta	16 rv	x これらのレジスタは、画素に解像されつつあるビデオの水平及び垂直次元を保持する。 A. 14.2 参照
horiz_macroblocks vert_macroblocks	16 rv	x これらのレジスタは、マクロブロックに解像されつつあるビデオの水平及び垂直次元を保持する。 A. 14.2 参照

図A. 14.2 ビデオデマックスピクチャ構造レジスタ

【0815】

【表103】

レジスタ名	ビット数	注
max_h max_v	2 rv	x これらのレジスタは、ブロックで示されるマクロブロックの幅及び高さを保持する（16×8画素）。max_hから3まで、1から4ブロック幅での幅/高さを示す。 A. 14.2 参照
max_component_id	2 rv	x 0から3までは、1から4までの異なるビデオ成分が表す順序を示しつつあることを示す。 A. 14.2 参照
NI	8 rv	x JPEG符号化レシジョン時、このレジスタはパラメータNI（フレーム内イメージ成分の値）を保持する。
blocks_h_0 blocks_h_1 blocks_h_2 blocks_h_3 blocks_v_0 blocks_v_1 blocks_v_2 blocks_v_3	2 rv	x 4色成分の各々に対して、レジスタ blocks_v_n は、成分ID n の色成分に属するブロック数、マクロブロック内の水平及び垂直に保持する。 A. 14.2 参照
te_0 te_1 te_2 te_3	2 rv	x レジスタ te_n によって保持される2ビット値は、成分ID n を持つデータ流に属してこの変換子化変を処理するべきかを示す。

図A. 14.2 ビデオデマックスピクチャ構造レジスタ

A. 14. 1. 1 「レジスタのローディング、及びトークンの生成」

動画デマルチプレクサ内の多数のレジスタは、コード化されたピクチャ／動画データにおいて通常伝達されるパラメータに直接関係する値を保持する。例えば、水平画素レジスタは、MPEGシーケンスヘッダ情報、水平サ
40 サイズ、及びJPEGフレームヘッダパラメータ、Xに対応する。これらのレジスタは、当該コード化データが復号化される時、動画デマルチプレクサによってロードされる。同様に、これらのレジスタもトークンと関連す

る。

【0816】例えば、水平画素レジスタ、トークン、水平サイズと関連する。トークンは、コード化データが復号化される時（又は、直後も）、動画デマルチプレクサによって生成される。同様に、トークンは、空間デコーダの入力に直接に供給可能である。この場合、トークンが持つ値は、当該トークンと関連した動画デマルチプレクサを構成する。

【0817】

【表104】

レジスタ名	ビット数	説明
dc_huff_0 dc_huff_1 dc_huff_2 dc_huff_3	2 rv	レジスタ dc_huff_n によって保持される2ビット値は、成分1D ₀ を用いたデータのDC係数を表すために、このハフマン係数表を用いるべき値を表す。同時に、ac_huff_n は、AC成分の振幅に準じて使用するべき値を表す。高圧縮JPGでは、通常より2つまでのハフマン表を必要とする。実現された値は0及び1だけである。
ac_huff_0 ac_huff_1 ac_huff_2 ac_huff_3	2 rv	
dc_bits_0(15:0) dc_bits_1(15:0) ac_bits_0(15:0) ac_bits_1(15:0)	2 rv 2 rv	これらの各々は、16ビットの8ビット値の表である。これらは、2つのDC及び2つのACハフマン表に属する記号の一部を対応するBITS表（JPEGハフマン表参照）を表す。 セクションA.14.3.1 参照
dc_huffval_0(15:0) dc_huffval_1(15:0)	2 rv	これらの各々は、16ビットの8ビット値の表である。これらは、2つのDCハフマン表に属する記号の一部を対応するHUFFVAL表（JPEGハフマン表参照）を表す。 セクションA.14.3.1 参照
ac_huffval_0(161:0) ac_huffval_1(161:0)	2 rv	これらの各々は、162ビットの8ビット値の表である。これらは、2つのACハフマン表に属する記号の一部を対応するHUFFVAL表（JPEGハフマン表参照）を表す。 セクションA.14.3.1 参照
dc_zeros_0 dc_zeros_1 ac_eob_0 ac_eob_1 ac_zrl_0 ac_zrl_1	8 rv 8 rv 8 rv	これらの8ビットレジスタは、しばしば使用されるJPEG VLCs の振幅を記憶するための（特殊な場合）に用意する値を保持する。 DC係数の dc_zeros - magnitude は0である。 ac_eob - end of block ac_zrl - run of 16 zeros

表A.14.3 ビデオデマックスハフマン表レジスタ

【0818】

【表105】

レジスタ名	ビット数	注
buffer_size	10 rw	このレジスタは、デコードコードに必要なVBVバッファのサイズを示す値を持つMP EGの領域に於いてロードされる。この値はデコードチップによって使用されない。但し、保持される値は、コード化データバッファの領域に於いてユーザソフトウェアに於き、及びデコードチップがMP EGデータファイルの領域が写像されるべき領域に於いて利用可能である。
pel_aspect	4 rw	このデコードは、垂直アスペクトレシオを示す値を持つMP EGデータの領域に於いてロードされる。値は、MP EGによって定義されるレベルにインデックスとして用いられる4ビット整数である。この値の定義に関してはMP EG規格を参照のこと。この値は、デコードチップによって使用されない。但し、保持される値は、ディスプレイ又は出力デバイスの領域に於いて、ユーザソフトウェアに於いて利用可能である。
bit_rate	18 rw	このレジスタは、コード化データレートを示す値を持つMP EGデータの領域に於いて、ロードされる。この値の定義に関しては、MP EG規格を参照のこと。この値は、デコードチップによって使用されない。但し、保持される値は、デコードスタートアップレジスタの領域に於いて、ユーザソフトウェアに於いて利用可能である。
pic_rate	4 rw	この値は、ピクチャレートを示す値を持つMP EGデータの領域に於いて、ロードされる。この値の定義に関しては、MP EG規格を参照のこと。この値は、デコードチップによって使用されない。但し、保持される値は、ディスプレイ又は出力デバイスの領域に於いて、利用可能である。
constrained	1 rw	このレジスタは、コード化データがMP EGによって制限されたパラメータに属するかどうかを示すために、MP EGデータ領域に於いて、ロードされる。このフラグの定義に関しては、MP EG規格を参照されたい。この値は、デコードチップによって使用されない。但し、保持される値は、デコードチップがMP EGデータファイルの領域が写像されるべき領域に於いて、ユーザソフトウェアに於いて利用可能である。

図A.14.4 4つのビデオマックスレジスタ

【0819】

【表106】

レジスタ名	ビット数	注																
picture_type	2 rv	MPEG作動期間中、このレジスタは、画面を1つあるピクチャのピクチャタイプを保持する。																
A_261_pic_type	8 rv	<p>このレジスタは、H.261データの取得に際してロードされ、ピクチャフォーマットに該当する情報を保持する。</p> <div style="text-align: center;"> <table border="1" style="margin: 10px auto;"> <tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr> <tr><td>r</td><td>r</td><td>s</td><td>0</td><td>r</td><td>q</td><td>r</td><td>r</td></tr> </table> </div> <p>フラグ:</p> <ul style="list-style-type: none"> 4: スプライトスクリーンインジケータ 6: ドキュメントカメラ 7: フリースピクチャリリース <p>この値は、デコーダチップによって使用されない。もし、両端は <i>her12_bits</i>、<i>ver1_bits</i> 及びディスプレイ又は出力デバイスと接続する際に使用されなければならない。</p>	7	6	5	4	3	2	1	0	r	r	s	0	r	q	r	r
7	6	5	4	3	2	1	0											
r	r	s	0	r	q	r	r											
broken_closed	2 rv	<p>MPEG作動期間中、このレジスタは、画面中のピクチャグループに対し、<i>broken_link</i> 及び <i>closed_gop</i> を保持する。</p> <div style="text-align: center;"> <table border="1" style="margin: 10px auto;"> <tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr> <tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>c</td><td>b</td></tr> </table> </div> <p>フラグ:</p> <ul style="list-style-type: none"> c: <i>closed_gop</i> 	7	6	5	4	3	2	1	0	r	r	r	r	r	r	c	b
7	6	5	4	3	2	1	0											
r	r	r	r	r	r	c	b											

図A.14.4 他のビデオデマックスレジスタ

【0820】

【表107】

レジスタ名	ビット数	注 記																
prediction_mode	8 rw	<p>MPEG2 H.261 規格の中で、このレジスタは、予測モードの先行値を使用する。</p> <div style="text-align: center;"> <table border="1" style="margin: 0 auto;"> <tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr> <tr><td>r</td><td>r</td><td>r</td><td>b</td><td>r</td><td>b</td><td>r</td><td>r</td></tr> </table> </div> <p>フラグ:</p> <p>b: H.261 スープファイルタポットモード</p> <p>r: 後方ベクトル予測リセット</p>	7	6	5	4	3	2	1	0	r	r	r	b	r	b	r	r
7	6	5	4	3	2	1	0											
r	r	r	b	r	b	r	r											
vbv_delay	16 rw	<p>このレジスタは、解読開始前の最小スタートアップ遅延を示す値を保持するMP EGデータの解読に際してロードされる。</p> <p>この値の定義については、MPEG規格を参照のこと。</p> <p>この値は、デコーダチップにより使用されない。もし、保持される場合は、デコーダスタートアップレジスタの値に等しい。ユーザソフトウェアによって書き換え可能である。</p>																
pic_number	8 rw	<p>このレジスタは、ビデオデマックスによって現在解読されているピクチャ用ピクチャ番号を保持する。この番号は、このピクチャが解読した時にスタートコードによって生成される。ピクチャ番号については図A.11.2 参照。</p>																
dummy_test_picture	1 rw	<p>これらのレジスタは、トップレベルにおいてのみ有効である。</p> <p>図A.14.1 参照</p>																
field_info	1 rw																	
continue	1 rw																	
rcm_revision	8 rw																	
coding_standard	2 rw	<p>このレジスタは、ビデオデマックスの動作モードを指定するために、CODING_STANDARDによってロードされる。</p> <p>セクションA21.1 参照</p>																

図A.14.4 他のビデオデマックスレジスタ

【0821】

30 【表108】

レジスタ名	ビット数	注 記
restart_interval	8 rw	このレジスタは、解読開始前の最小スタートアップ遅延を示す値を用いるJPE Gデータの解読に際してロードされる。 この値の定義については、MPEG規格を参照。

図A.14.4 他のビデオデマックスレジスタ

【0822】

【表109】

レジスタ	トークン	規格	コメント
component_name_0	COMPONENT_NAME	JPEG	コード化データにおいて
		MPEG	規格において使用されず
		H.261	
horiz_pels	HORIZONTAL_SIZE	MPEG	コード化データにおいて
vert_pels	VERTICAL_SIZE	JPEG	
		H.261	ピクチャタイプから自動的に発生
horiz_macroblocs	HORIZONTAL_MBS	MPEG	別部ソフトウェアが生成しなければならない。サンプリング周波数は規格によって規定される。
vert_macroblocs	VERTICAL_MBS	JPEG	
		H.261	ピクチャタイプから自動的に発生
max_h	DEFINE_MAX_SAMPLING	MPEG	別部ソフトウェアが生成しなければならない。サンプリング周波数は規格によって規定される。
max_v		JPEG	コード化データにおいて
		H.261	4:2:0 ビデオに対して自動的に生成される。

表A.14.5 レジスタ対トークン相互参照

[0823]

【表110】

レジスタ	トークン	規格	コメント
max_component_id	MAX_COMP_ID	MPEG	別部ソフトウェアが生成しなければならない。サンプリング周波数は規格によって規定される。
		JPEG	コード化データにおいて
		H.261	4:2:0 ビデオに対して自動的に生成される。
tc_0	JPEG_TABLE_SELECT	JPEG	コード化データにおいて
tc_1		MPEG	規格において使用されず
tc_2		H.261	
tc_3			
blocks_h_0	DEFINE_SAMPLING	MPEG	別部ソフトウェアが生成しなければならない。サンプリング周波数は規格によって規定される。
blocks_h_1			
blocks_h_2			
blocks_h_3		JPEG	コード化データにおいて
blocks_v_0		H.261	4:2:0 ビデオに対して自動的に生成される。
blocks_v_1			
blocks_v_2			
blocks_v_3			
dc_buff_0	スキャンヘッダデータ	JPEG	コード化データにおいて
dc_buff_1	MPEG_BLOCK_TABLE	MPEG	別部ソフトウェアが生成しなければならない。
dc_buff_2		H.261	規格において使用されず
dc_buff_3			
ac_buff_0	スキャンヘッダデータ	JPEG	コード化データにおいて
ac_buff_1		MPEG	規格において使用されず
ac_buff_2		H.261	
ac_buff_3			

表A.14.5 レジスタ対トークン相互参照

[0824]

【表111】

レジスタ	トークン	規格	コメント
dc_bits_0[16:0] dc_bits_1[15:0]	in DATA Token following DMT_MARKER Token	JPEG	コード化データにおいて
dc_huffval_0[11:0] dc_huffval_1[11:0]		MPEG	新ソフトウェアが規格しなればならぬ
dc_tsasss_0 dc_tsasss_1		H.261	規格において使用されず
ac_bits_0[15:0] ac_bits_1[15:0]	in DATA Token following DMT_MARKER Token	JPEG	コード化データにおいて
ac_huffval_0[181:0] ac_huffval_1[181:0]		MPEG	規格において使用されず
ac_scb_0 ac_scb_1 ac_trf_0 ac_trf_1		H.261	規格において使用されず
buffer_size	VBV_BUFFER_SIZE	MPEG	コード化データにおいて
		JPEG	規格において使用されず
		H.261	
pel_aspect	PEL_ASPECT	MPEG	コード化データにおいて
		JPEG	規格において使用されず
		H.261	
bit_rate	BIT_RATE	MPEG	コード化データにおいて
		JPEG	規格において使用されず
		H.261	
pic_rate	PICTURE_RATE	MPEG	コード化データにおいて
		JPEG	規格において使用されず
		H.261	
constrained	CONSTRAINED	MPEG	コード化データにおいて
		JPEG	規格において使用されず
		H.261	
picture_type	PICTURE_TYPE	MPEG	コード化データにおいて
		JPEG	規格において使用されず
		H.261	

表 A.14.5 レジスタ対トークン対応表

[0825]

[表112]

レジスタ	トークン	規格	コメント
broken_closure	BROKEN_CLOSED	MPEG	コード化データにおいて
		JPEG	規格において使用されず
		H.261	
prediction_mode	PREDICTION_MODE	MPEG	コード化データにおいて
		JPEG	規格において使用されず
		H.261	
h_261_pic_type	PICTURE_TYPE (H.261規格)	MPEG	同値
		JPEG	
		H.261	コード化データにおいて
vbv_delay	VBV_DELAY	MPEG	コード化データにおいて
		JPEG	規格において使用されず
		H.261	
pic_number	Carried by: PICTURE_START	MPEG	スタートコード検出部により生成される。
		JPEG	
		H.261	
coding_standard	CODING_STANDARD	MPEG	新ソフトウェアは出荷によりスタートコードに
		JPEG	おいて検出される。
		H.261	

表 A.14.5 レジスタ対トークンの対応表

A. 14.2 「ピクチャの構造」

本発明におけるピクチャの次元は、空間デコーダに対して2つの異なる単位、即ち、ピクセル及びマクロブロックとして記述される。JPEG及びMPEGは双方共、ピクチャ次元をピクセルとして伝達する。ピクセルとし

て次元を伝達することにより、有効なデータを含むバッファの領域を決定する。この場合の当該領域は全バッファサイズより小さくても差し支えない。マクロブロックとして次元を伝達することにより、デコーダによって必要とされるバッファのサイズを決定する。マクロブロッ

ク次元は、ピクセルの次元からユーザによって求められなければならない。この情報と関連している空間デコーダレジスタを次に示す、即ち、水平画素、垂直画素、水平マクロブロック、及び垂直マクロブロックである。

【0826】空間デコーダレジスタ、ブロック数 h_n 、 v_n 、 max_h 、 max_v 、及び $max_component_id$ は、マクロブロックの合成 (JPEGにおけるコード化最小単位) を指定する。各々は、0から3までの範囲内に値を保持することの出来る

1つの2ビットのレジスタである。 $max_component_id$ を除く全てのレジスタは、1から4までのブロックカウントを指定する。例えば、レジスタ max_h が1を保持する場合、マクロブロックの幅は2ブロックである。同様に、 $max_component_id$ は、関係する異った色成分の数を指定する。

【0827】

【表113】

	2:1:1	4:2:2	4:2:0	1:1:1
max_h	1	1	1	0
max_v	0	1	1	0
$max_component_id$	2	2	2	2
$blocks_h_0$	1	1	1	0
$blocks_h_1$	0	0	0	0
$blocks_h_2$	0	0	0	0
$blocks_h_3$	x	x	x	x
$blocks_v_0$	0	1	1	0
$blocks_v_1$	0	1	0	0
$blocks_v_2$	0	1	0	0
$blocks_v_3$	x	x	x	x

表 A.14.6 各種マクロブロックフォーマット用コンフィギュレーション

A. 14. 3 「ハフマン表」

A. 14. 3. 1 「JPEGスタイルハフマン表記述」

本発明において、ハフマン表記述は、エンコーダとデコーダとの間で表記述を伝達するために、JPEGによつて使われるフォーマットを介して空間デコーダに提供される。各表記述には2つのエレメント、即ち、BITS及びHUFFVALがある。ユーザーは、表がコード化される方法についての完全な記述に関して、JPEG仕様に案内される。

【0828】A. 14. 3. 1. 1 「BITS」

BITSは、VLCの各長さを用いて幾つの異なる記号が符号化されるかについて記述する値を示す表である。各エントリは1つの8ビット値である。JPEGは、VLCの長さとして最大16ビットまでを許可する。従つて、各表に16個のエントリがある。

【0829】BITS [0] は、幾つの異なる1ビットVLCが存在するかを記述し、他方、BITS [1] は、幾つの異なる2ビットVLCが存在するかを記述する、等々。

【0830】A. 14. 3. 1. 2 「HUFFVAL」

HUFFVALは、VLCの長さの増大順に整列された8ビットデータ値の表である。この表のサイズは、VLC

Cによってコード化可能な異なる記号の数に依存する。

【0831】JPEG仕様は、ハフマンコード化表がこのフォーマットに対してコード化又は復号化される方法について更に詳細に記述する。

【0832】A. 14. 3. 1. 3 「トークンによるコンフィギュレーション」

JPEGビットストリームにおいて、AC及びDC係数をコード化するために使われるハフマン表の記述にはDHTマーカが先行する。スタートコード検出器がDHTマーカを認識する場合、検出器は、DHTマーカトークンを生成し、そして、次に後続するデータトークン内にハフマン表記述を配置する (A. 11. 3. 4参照)。

【0833】空間デコーダ内におけるAC及びDC係数ハフマン表の構成は、空間デコーダの入力にデータ及びDHTマーカトークンを供給することによって達成可能であり、同時に、空間デコーダはJPEGオペレーションに対して構成される。このメカニズムは、MPEGオペレーションのために必要とされるDC係数ハフマン表を構成するために使用出来るが、ただし、表がダウンロードされると同時に、空間デコーダの符号化規格はJPEGにセットされなければならない。

【0834】

【表114】

E	7	6	5	4	3	2	1	0	トークン名	
1	0	0	0	1	0	1	0	1	CODING_STANDARD	
0	0	0	0	0	0	0	0	1	I=JPEG	
0	0	0	0	1	1	1	0	0	DHT_MARKER	
1	0	0	0	0	0	1	X	X	DATA	
1	t	t	t	t	t	t	t	t	TH-ハフマン表がロードされるべきであることを示す値。JPEGは4つの表がダウンロードされることを可能にする。 値0x00及び0x01は、表0及び1をコード化するDC係数を指定する。値0x10及び0x11は、表0及び1をコード化するAC係数を指定する。 LI-BITS情報を持つ16ワード Vj-HUFFVAL情報を持つワード（ワード数は異なる記号の数に依存する）。 e・これがDATAトークンのエンドであるか又は1である場合には、拡張ビットは0。もう1つの表記述が同一DATAトークンに含まれる場合には、拡張ビットは1。	このシーケンスは複数の表を、単一トークンにおいて記述可能にする。
1	n	n	n	n	n	n	n	n		
1	n	n	n	n	n	n	n	n		
1	n	n	n	n	n	n	n	n		

表A.14.7 トークンを介したハフマン表のコンフィギュレーション

A. 14. 1. 4 「MPIによるコンフィギュレーション」

AC及びDC係数ハフマン表は、同様に、MPIを介してレジスタに直接記入することが可能である。表104を参照されたい。

【0835】*レジスタdc bits0 [15:0] 及びdc bits1 [15:0] は、表0x00及び0x01用のBITS値を保持する。

【0836】*レジスタac bits0 [15:0] 及びac bits1 [15:0] は、表0x10及び0x11用のBITS値を保持する。

【0837】*レジスタdc huffval0 [11:0] 及びdc huffval1 [11:0] は、表0x00及び0x01用のHUFFVAL値を保持する。

【0838】*レジスタac huffval0 [161:0] 及びac huffval1 [161:0] は、表0x10及び0x11用のHUFFVAL値を保持する。

【0839】A. 14. 4 「異なる規格用構成」
 動画デマルチプレクサは、MPEG、JPEG、及びH. 261の必要条件をサポートする。符号化規格は、スタートコード検出器によって生成されたCODING STANDARDトークンによって自動的に構成される。

【0840】A. 14. 4. 1 「H. 261ハフマン表」

H. 261を復号化するために必要とされる全てのハフ

マン表は、空間デコーダ内のROMに保持され、更に詳細には、動画デマルチプレクサのパーザステートマシン内に保持されるので、ユーザーの介入を必要としない。

【0841】A. 14. 4. 2 「H. 261ピクチャの構造」

H. 261は、わずか2つのピクチャフォーマット、即ち、CIF及びQCIFをサポートするものとして定義される。使用中のピクチャフォーマットは、ビットストリームのPTYPEセクションにおいて送信される。このデータは、空間デコーダによって復号化されると、H. 261ピクチャタイプレジスタ及びピクチャタイプトークン内に配置される。更に、全てのピクチャ及びマクロブロック構成レジスタは自動的に構成される。

【0842】様々なレジスタ内の情報は、それらの関連トークン内に配置され表109～表112参照）、そして、これによって、他のデコーダチップ（例えば、時間デコーダ）が正しく構成されていることを保証する。

【0843】A. 14. 4. 3 「MPEGハフマン表」

MPEGを復号化するために必要なハフマンコード化表の大多数は、空間デコーダ内のROM内に保持されるので（パーザステートマシン内であることを再度述べておく）、ユーザの介入を必要としない。イントラルマクロブロックのDC係数を復号化するために必要な表は例外である。1つはクロマ用、そして、他の1つはルマ用として2つの表が必要とされる。これらの表は、復号化が始まる前に、ユーザソフトウェアによって構成されなければならない。

【0844】

【表115】

マクロブロックの構造	CIF/	ピクチャの構造	CIF	CCIF
	CCIF			
max_h	1	horiz_pels	352	176
max_v	1	vert_pels	288	144
max_component_id	2	horiz_macroblocks	22	11
blocks_h_0	1	vert_macroblocks	18	9
blocks_h_1	0			
blocks_h_2	0			
blocks_v_0	1			
blocks_v_1	0			
blocks_v_2	0			

表A.14.8 H.261用自動設定

表117は、空間デコーダ内のDC係数ハフマン表を構成するために必要とされるトークンのシーケンスを示す。その代りに、MPIを介してこの情報をレジスタに記入することにより、同じ結果が得られる。

【0845】レジスタdc_huff_nは、各色成分と共にどのDC係数ハフマン表を使用するかを制御する。

表116は、MPEGオペレーション用としてどのように構成されなければならないかを示す。これは、MPIを介するか、或いは、MPEGDCHテーブルトークンを用いることによって直接行うことが可能である。

【0846】

【表116】

dc_huff_0	0
dc_huff_1	1
dc_huff_2	1
dc_huff_3	x

表A.14.9 MPIを介したMPEG DCハフマン表の選択

【0847】

30 【表117】

E	[7:0]	ト ー ク ン 名
1	0x15	CODING_STANDARD
0	0x01	I = JPEG
0	0x1C	DHT_MARKER
1	0x04	DATA (任意の色成分であって差支えない。本例は0が用いられている)
1	0x00	0はこのハフマン表がDC係数コード化表0であることを示す。

表A.14.10 MPEG DCハフマン表コンフィギュレーション

【0848】

【表118】

E [7:0]		ト ー ク ン 名
1	0xC0	合計9の異なるVLCを記述するBITS情報を持つ16ワード
1	0x02	
1	0x03	2.2 ビットコード
1	0x01	3.3 ビットコード
1	0x01	1.4 ビットコード
1	0x01	1.5 ビットコード
1	0x01	1.6 ビットコード
1	0xC0	1.7 ビットコード
1	0xC0	トークン以外のMPLIを介して構成する場合には、これらの値は dc_bits_0
1	0xC0	[15:0]レジスタに記入される。
1	0xC0	
1	0xC0	
1	0xC0	
1	0xC0	
1	0xC0	
1	0x01	HUFFVAL情報を持つ9ワード
1	0x02	
1	0xC0	トークン以外のMPLIを介して構成する場合には、これらの値は
1	0x03	dc_huffval_0[11:0]レジスタに記入される。
1	0x04	
1	0x05	
1	0x06	
1	0x07	
0	0x08	

表 A.14.10 MPEG DCハフマン表コンフィギュレーション

[0849]

【表119】

E	[7:0]	ト ー ク ン 名
0	0x1C	DHT_MARKER
1	0x04	DATA (任意の色成分であって空支えない。本例では0が用いられる。)
1	0x01	1はこのハフマン表がDC係数コード化表1であることを示す。
1	0x00	合計9の異なるVLCを記述するBITS情報を持つ16ワード
1	0x03	
1	0x01	3.2 ビットコード
1	0x01	1.3 ビットコード
1	0x01	1.4 ビットコード
1	0x01	1.5 ビットコード
1	0x01	1.6 ビットコード
1	0x01	1.7 ビットコード
1	0x00	1.8 ビットコード
1	0x00	トークン以外のMPLを介して構成する場合には、これらの値は
1	0x00	dc_bits_1[15:0] レジスタに記入される。
1	0x00	
1	0x00	
1	0x00	
1	0x00	
1	0x00	HUFFVAL情報を持つ9ワード
1	0x01	トークン以外のMPLを介して構成する場合には、これらの値は
1	0x02	dc_huffval_1[11:0] レジスタに記入される。
1	0x03	
1	0x04	
1	0x05	
1	0x06	
1	0x07	
0	0x08	
1	0x04	MPEG_DCH_TABLE
0	0x00	表0が成分0用に用いられるように構成すること。
1	0x05	MPEG_DCH_TABLE
0	0x01	表1が成分1用に用いられるように構成すること。

表A.14.10 MPEG DCハフマン表コンフィギュレーション

【0850】

【表120】

E	[7:0]	ト ー ク ン 名
1	0x06	MPEG_DCH_TABLE
0	0x01	表1が成分2用に用いられるように構成すること。
1	0x15	CODING_STANDARD
0	0x02	2 = JPEG

表A.14.10 MPEG DCハフマン表コンフィギュレーション

A. 14. 4. 4 「MPEGピクチャの構造」
MPEG用に定義されたマクロブロックの構成は、H. 261によって用いられる構造と同じである。ピクチャの次元はコード化データにコード化される。

【0851】規格4:2:0オペレーションに対する場合、マクロブロックの特性は、表115に指示されたように構成されなければならない。これは、指示されたようにレジスタに記入するか、或いは、等値のトークンを

空間デコードの入力に供給することによって行うことが可能である表109～表112参照。

【0852】ピクチャ次元を構成するために用いられる方法は、アプリケーションに依存する。復号化開始前にピクチャフォーマットが既知である場合には、表115に示されるピクチャ構成レジスタは、適切な値を用いて初期化可能である。その代りに、ピクチャ次元は、コード化データから復号化することが可能であり、そして、

空間デコーダを構成するために使用することが可能である。この場合、ユーザは、パーザエラーMPEGシーケンスをサポートしなければならない。A. 14. 8 「MPEGシーケンス層における変化」参照。

【0853】A. 14. 4. 5 「JPEG」

基底線JPEG内には、デコーダを操作するために必要な制御ソフトウェアの複雑さを大幅に変更する多数のエンコーダオプションがある。一般に、空間デコーダは、次に示す条件に適合する場合に必要なとされるサポートが最小限であるように設計されている。

* フレーム当たりの色成分の数が5未満であること ($Nf \leq 4$)。

【0854】A. 14. 4. 6 「JPEGハフマン表」

更に、JPEGは、ハフマンコード化表がダウンロードされることを可能にする。これらの表は、係数を記述するVLCを復号化する場合に用いられる。DC係数を復号化するために走査当たり2つの表が許可され、そして、AC係数用にも2つの表が許可される。

【0855】3つの異なるタイプのJPEGファイルがある、即ち、交換フォーマット、圧縮されたイメージデータ用短縮フォーマット、及び表データ用短縮フォーマットである。交換フォーマットファイルには、圧縮されたイメージデータ及びイメージデータを復号化するために必要な全ての表（ハフマン、量子化等）の定義の両者が含まれる。短縮されたイメージデータフォーマットファイルは、表定義を省略する。短縮された表フォーマットファイルは表の定義のみを含む。

【0856】空間デコーダは、3つ全てのフォーマットを受け入れる。ただし、必要とされる全ての表が定義済みである場合には、短縮されたイメージデータファイルのみが復号化可能である。他の2つのタイプのJPEGファイルのうちのどちらかを介してこの定義は実施可能であり、或は、その代りに、ユーザソフトウェアによって表をセットアップすることが可能である。

【0857】各々の走査が異なる1組のハフマン表を使用する場合、各スキャンの前に表の定義がコード化データ内に（エンコーダによって）配置される。これらの定義は、当該走査および後続する走査期間中に使用するために、空間デコーダによって自動的にロードされる。

【0858】ハフマン復号化の性能を改良するためには、特定の共通に使用される記号が特別にケースされる。これらの記号は、大きさ0のDC係数、ブロックAC係数のエンド、及び16のゼロAC係数のランである。これら特別なケース用の値は、該当するレジスタに記入されなければならない。

【0859】A. 14. 4. 6. 1 「表の選択」
レジスタdc_huff_n及びac_huff_nは、どの色成分と共にどのAC及びDC係数ハフマン表を用いるかを制御する。JPEGオペレーション期間

中、これらの関係は、走査ヘッダシンタックスのTDJ及びTajフィールドによって定義される。

【0860】A. 14. 4. 7 「JPEGピクチャの構造」

空間デコーダによってサポートされた基底線JPEG復号化には明白な2つのレベルがある、即ち、フレーム当たり4成分まで ($Nf \leq 4$) のレベル、及びフレーム当たり4成分より大きい ($Nf > 4$) レベルである。 $Nf > 4$ を使用する場合には、必要とされる制御ソフトウェアは更に複合になる。

【0861】A. 14. 4. 7. 1 「 $Nf \leq 4$ 」

JPEGフレームヘッダに含まれるフレーム成分仕様パラメータは、これらの復号化に際してマクロブロック構成レジスタを構成する表115参照）。4つの異なる色成分を復号化するために必要な全ての仕様が定義されるので、ユーザの介入は必要としない。

【0862】JPEGによって提供されるオプションの詳細に関しては、JPEG仕様を研究されたい。同様に、JPEGピクチャフォーマットに関してはセクションA. 16. 1に簡単に記述されている。

【0863】A. 14. 4. 7. 2 「4より多い成分を持つJPEG」

空間デコーダは、最大256の異なる色成分（JPEGにより許可される最大数）を含むJPEGファイルの復号化が可能である。ただし、4成分より多い色成分を復号化しようとする場合には、付加的なユーザ介入が必要とされる。JPEGにおいては、あらゆる走査において最大限4成分までが許可されるに過ぎない。

【0864】A. 14. 4. 8 「非規格バリエーション」

既に述べたように、空間デコーダは、JPEG及びMPEGによって定義されたフォーマットよりも多いピクチャフォーマットをサポートする。

【0865】JPEGは、コード化単位が1走査当たり10ブロックよりも多いブロックを含まないように、最小コード化単位を制限する。空間デコーダは、ブロック数h_n、ブロック数v_n、max_h、及びmax_vによって記述可能な任意の最小コード化単位を処理することが出来るので、この限界は、空間デコーダには適用されない。

【0866】MPEGにおいては、4:2:0マクロブロックに関してのみ定義される表115参照）。ただし、空間デコーダは、3つの他の成分マクロブロック構造（例えば4:2:2）を処理可能である。

【0867】A. 14. 5 「動画イベント及びエラー」

動画デマルチプレクサは、2つのタイプのイベント、即ち、パーザイベント及びハフマンイベントを生成可能である。イベント及び割込みを扱う方法の記述に関しては、A. 6. 3「割込み」を参照されたい。

【0868】A. 14. 5. 1 「ハフマンイベント」
ハフマンイベントは、ハフマンデコードによって生成される。ハフマンイベント及びハフマンマスクにおいて指示されるイベントは、割込みが生成されるかどうかを決定する。ハフマンマスクが1にセットされる場合には、割込みが生成され、そして、ハフマンデコードが停止する。レジスタハフマンエラーコード [2 : 0] は、イベントの原因を示す値を保持する。

【0869】割り込みをサービスした後でハフマンイベントに1が記入される場合には、ハフマンデコードは、エラーからの回復を試行する。同様に、ハフマンマスクが0にセットされた場合には（割込みをマスクし、そして、ハフマンデコードを停止させない）、ハフマンデコードは、エラーからの回復を自動的に試行する。

【0870】A. 14. 5. 2 「パーザイベント」
パーザイベントは、パーザによって生成される。このイベントは、パーザイベントにおいて指示される。それ以降においては、割込みが生成されるかどうかは、パーザマスクによって決定される。パーザマスクが1にセットされている場合には、割込みが生成され、そして、パーザは停止する。レジスタパーザエラーコード

[7 : 0] は、イベントの原因を示す値を保持する。

【0871】割込みをサービスした後で、ハフマンイベントに1が記入される場合には、ハフマンデコードは、エラーからの回復を試行する。同様に、ハフマンマスクが0にセットされた場合には（割込みをマスクし、そして、ハフマンデコードを停止させない）、ハフマンデコードは、エラーからの回復を自動的に試行する。

【0872】割り込みをサービスした後で、パーザイベントに1を記入する場合には、パーザは、オペレーションを再開する。イベントがビットストリームエラーを指示した場合には、動画デマルチプレクサは、エラーからの回復を試行する。

【0873】パーザマスクが0にセットされた場合には、パーザは、そのイベントビットをセットするが、割込み又は停止は生成しない。パーザは、オペレーションを継続し、そして、エラーからの自動的な回復を試行する。

【0874】

【表121】

huffman_error_code			記 述
[2]	[1]	[0]	
0	0	0	エラー無し。このエラーは、正常作動中に起きてはならない。
x	0	1	16ビット以内のVLCにおいて端末コードが発見できなかった。
x	1	0	トークンが期待される場合に発見できた直列データ
x	1	1	直列データが期待される場合に発見できたトークン
1	x	x	1つの単一ブロック用の64係数よりも多くを記述する情報が解読された。この情報はビットストリームエラーを示す。ビデオデマックスによって出力されるブロックは、64係数だけを含む。

表A.14.11 ハフマンエラーコード

【0875】

【表122】

parser_error_code(7:0)	記 述
0x00	ERR_NO_ERROR パーザエラー発生せず。このイベントは正常作動期間中は発生しない。
0x10	ERR_EXTENSION_TOKEN EXTENSION_DATAトークンが、パーザにより検出された。 このトークンの検出は、拡張データを含むDATAトークンに先行する ものである。A.14.6 参照
0x11	ERR_EXTENSION_DATA EXTENSION_DATAトークンの検出に続いて、拡張データを含む DATAトークンが検出された。A.14.6 参照
0x12	ERR_USER_TOKEN USER_DATAトークンが、パーザにより検出された。このトークン の検出は、ユーザデータを含むDATAトークンに先行する。A.14.6 参照
0x13	ERR_USER_DATA USER_DATAトークンの検出に続いて、ユーザデータを含むDATA トークンが検出された。A.14.6 参照
0x20	ERR_PSPARE H.261 PSARE情報が検出された。A.14.7 参照

表A.14.12 パーザエラーコード (1/5)

【0876】

【表123】

parser_error_code(7:0)	記 述
0x21	ERR_GSPARE H.261 GSPARE情報が検出された。A.14.7 参照
0x22	ERR_PTYPE H.261ピクチャタイプの値が変更した。レジスタ h_261_pic_type は、新 しい値を知るために検出可能である。
0x30	ERR_JPEG_FRAME
0x31	ERR_JPEG_FRAME_LAST
0x32	ERR_JPEG_SCAN ピクチャサイズ又はNs 変更済
0x33	ERR_JPEG_SCAN_COMP 成分変換!
0x34	ERR_DNL_MARKER
0x40	ERR_MPEG_SEQUENCE MPEGシーケンス層において伝送されたパラメータの1つが変更した。 A.14.8 参照
0x41	ERR_EXTRA_PICTURE MPEG extra_information_pictureが検出された。A.14.7 参照
0x42	ERR_EXTRA_SLICE MPEG extra_information_sliceが検出された。A.14.7 参照
0x43	ERR_VBV_DELAY 新しいMPEGビデオシーケンス内の最初のピクチャに関するVBV_ DELAYパラメータが、ビデオデマックスによって検出された。遅延の 新しい値は、レジスタ vbv_delayにおいて利用可能である。新しいシー ケンス最初のピクチャは、シーケンスエンド、FLUSH又はリセット後の 最初のピクチャとして定義される。
0x80	ERR_SHORT_TOKEN 不正に形成されたトークンが検出された。このエラーは、正常作動中、 発生してはならない。

表A.14.12 パーザエラーコード (2/5)

【0877】

50 【表124】

parser_error_code[7:0]	記 述
0x90	ERR_H_261_PIC_END_UNEXPECTED H.261作動期間中にピクチャの終端が、予期しない位置に現れた。これは、コード化データ内のエラーを示すと検出される。
0x91	ERR_GN_BACKUP H.261作動期間中に予期したより小さいグループ番号を持つブロックグループが現れた。これは、コード化データ内のエラーを示すと検出される。
0x92	ERR_GN_SKIP_GOB H.261作動期間中に予期したより大きいグループ番号を持つブロックグループが現れた。これは、コード化データ内のエラーを示すと検出される。
0xA0	ERR_NBASE_TAB JPEG作動期間中に基座級JPEGによりサポートされないハフマン表へのダウンロードが試みられた (基座級JPEGはエントロピコード化に対して表0及び1のみをサポートする。)
0xA1	ERR_QUANT_PRECISION JPEG作動期間中に基座級JPEGによりサポートされない量子化表へのダウンロードが試みられた (基座級JPEGは量子化表における8ビット精度をサポートするだけである。)
0xA2	ERR_SAMPLE_PRECISION JPEG作動期間中に基座級JPEGによりサポートされるより高度のサンプル精度の指定が試みられた (基座級JPEGは8ビット精度をサポート)
0xA3	ERR_NBASE_SCAN JPEG定数ヘッダパラメータSs, Se, Ah 及びAl の1つ又はそれ以上が基座級JPEGによりサポートされない値にセットされる (基座級JPEGによりサポートされないスペクトル選択、及び/又は離散的近似を示す)
0xA4	ERR_UNEXPECTED_DNL JPEG作動期間中にフレーム最初の定数でない定数において、DNLマークが現れた。
0xA5	ERR_EOS_UNEXPECTED JPEG作動期間中に、予期しない場所にEOSマークが現れた。

表 A. 14. 12 パーザーエラーコード(3/5)

[0878]

[表125]

parser_error_code[7:0]	記 述
0x16	ERR_RESTART_SKIP JPEG作動期間中に、リスタートマークが予測しない場所に現れるか、又はリスタートマークが予測しない値である。リスタートマークが予測された時に現れない場合、ハフマンイベント（トークンが予測された場合に現れる直列データ）が生成される。
0x30	ERR_SKIP_INTRA MPEG作動期間中に、1より大きいマクロブロックアドレスインクリメントを持つマクロブロックが、イントラ(1)ピクチャ内に現れた。これは、不当であり、ビットストリームエラーを示すものと推測される。
0x31	ERR_SKIP_DINTRA MPEG作動期間中に、1より大きいマクロブロックアドレスインクリメントを持つマクロブロックが、DCだけの(D)ピクチャ内に現れた。これは不当であり、ビットストリームエラーを示すものと推測される。
0x52	ERR_BAO_MARKER MPEG作動期間中に、マークビットが予測した値を持たなかった。これはビットストリームエラーを示すものと推測される。
0x53	ERR_D_MBTYPE MPEG作動期間中にDCだけの(D)ピクチャ内に1以外のマクロブロックタイプを持つマクロブロックが現れた。これは不当であり、ビットストリームエラーを示すものと推測される。
0x54	ERR_D_MBEND MPEG作動期間中にDCだけの(D)ピクチャ内にマクロブロックビットの終端に0を持つマクロブロックが発見された。
0x55	ERR_SVP_BACKUP MPEG作動期間中に予測したより小さいスライス垂直位置を持つスライスが現れた。
0x56	ERR_SVP_SKIP_ROWS MPEG作動期間中に予測したより大きいスライス垂直位置を持つスライスが現れた。これはコード化データ内のエラーを示すものと推測される。
0x57	ERR_FST_MBA_BACKUP MPEG作動期間中に予測したより小さいマクロブロックアドレスを持つマクロブロックが現れた。これはコード化データ内のエラーを示すものと推測される。

表A.14.12 パーザーエラーコード(4/5)

【0879】

30 【表126】

parser_error_code[7:0]	記 述
0x58	ERR_FST_MBA_SKIP MPEG作動期間中に予測したより大きいマクロブロックアドレスを持つマクロブロックが現れた。これはコード化データ内のエラーを示すものと推測される。
0x59	ERR_PICTURE_END_UNEXPECTED MPEG作動期間中にPICTURE_ENDトークンが予測しない場所に現れた。これはコード化データ内のエラーを示すものと推測される。
0x5D...0xEF	内部テストプログラム用に予約済みのエラー
0xE0	ERR_TST_PROGRAM テストプログラム内において不可解に到着した。
0xE1	ERR_NO_PROGRAM テストプログラムがコンパイルされない場合
0xE2	ERR_TST_END テストエンド
0xF0...0xFF	予約済みエラー
0xFD	ERR_UCODE_ADDR ワードエンドがフォールオフする
0xF1	ERR_NOT_IMPLEMENTED

表A.14.12 パーザーエラーコード(5/5)

各規格は、定義されたパーザエラーコードの異なる部分
集合を使用する。

【0880】

【表127】

トークン名	MPEG	JPEG	H261
ERR_NO_ERROR	/	/	/
ERR_EXTENSION_TOKEN	/	/	
ERR_EXTENSION_DATA	/	/	
ERR_USER_TOKEN	/	/	
ERR_USER_DATA	/	/	
ERR_PSPARE			/
ERR_GSPARE			/
ERR_PTYPE			/
ERR_JPEG_FRAME		/	
ERR_JPEG_FRAME_LAST		/	
ERR_JPEG_SCAN		/	
ERR_JPEG_SCAN_COMP		/	
ERR_DNL_MARKER		/	
ERR_MPEG_SEQUENCE	/		
ERR_EXTRA_PICTURE	/		
ERR_EXTRA_SLICE	/		
ERR_VBV_DELAY	/		
ERR_SHORT_TOKEN	/	/	/
ERR_H261_PIC_END_UNEXPECTED			/
ERR_GN_BACKUP			/
ERR_GN_SKIP_GOB			/
ERR_NBASE_TAB		/	
ERR_QUANT_PRECISION		/	
ERR_SAMPLE_PRECISION		/	
ERR_NBASE_SCAN		/	
ERR_UNEXPECTED_DNL		/	
ERR_EOS_UNEXPECTED		/	
ERR_RESTART_SKIP		/	
ERR_SKIP_INTRA	/		
ERR_SKIP_DINTRA	/		
ERR_BAD_MARKER	/		

表A.14.13 パーザエラーコード及び種々の規格(1/2)

【0881】

【表128】

トークン名	MPEG	JPEG	H261
ERR_D_MBTYPE	/		
ERR_D_MBEND	/		
ERR_SVP_BACKUP	/		
ERR_SVP_SKIP_ROWS	/		
ERR_FST_MBA_BACKUP	/		
ERR_FST_MBA_SKIP	/		
ERR_PICTURE_END_UNEXPECTED	/		
ERR_TST_PROGRAM	/	/	/
ERR_NO_PROGRAM	/	/	/
ERR_TST_END	/	/	/
ERR_UCODE_ADDR	/	/	/
ERR_NOT_IMPLEMENTED	/	/	/

表A.14.13 パーザエラーコード及び種々の規格(2/2)

A. 14. 6 「ユーザデータ及び拡張データの受信」
MPEG及びJPEGは、ユーザデータ及び拡張データ
を埋め込むための同様のメカニズムを用いる。データ
は、スタート/マーカコードによって先行される。アプ
リケーションが当該データに関心を持たない場合には、
スタートコード検出器が、このデータを削除するように
構成可能である (A. 11. 3. 3参照)。

【0882】 A. 14. 6. 1 「データソースの識別」

パーザイベント、ERREXTENSIONトークン、
及びERRユーザトークンは、動画デマルチプレクサに
おけるEXTENSIONデータ、またはユーザデータ
トークンの到着を示す。これらのトークンがスタートコ
ード検出器によって生成された場合には (A. 11.

3. 3参照)、これらのトークンは、スタートコード検
出器にトークンを生成させたスタート/マーカコードの
値を持つ表92参照)。この値は、パーザ割込みをサポ
ートしている間に、rom revisionレジスタ

を読むことによって読み取り可能である。パーザイベントに1が記入されるまで動画デマルチプレクサは、停止状態を維持する(A. 6. 3、「割込み」参照)。

【0883】A. 14. 6. 2 「データの読取り」
EXTENSIONデータ、及びユーザデータトークンは、拡張データ又はユーザデータを持つデータトークンにより直ちに後続されるものと予測される。動画デマルチプレクサは、このデータトークンの到着によって、ERR EXTENSIONデータ又はERRユーザデータパーザイベントのいずれかを生成する。データトークンの第1のバイトは、割込みをサービスしている間に、rom revisionレジスタを読むことによって読み取り可能である。

【0884】動画デマルチプレクサレジスタの継続状態は、イベントがクリアされた後における行動態様を決定する。このレジスタが値0を保持する場合には、データトークンにおけるあらゆる残りのデータは動画デマルチプレクサによって消費され、そして、イベントは一切生成されない。継続が1にセットされている場合には、拡張の各バイト又はユーザデータが動画デマルチプレクサに到着するにつれてイベントが生成される。これは、データトークンが使い果たされるか、或いは、継続が0にセットされるまで継続する。

【0885】注記：

1) 拡張/ユーザデータの第1のバイトは、継続の状態に関係なく、rom revisionレジスタを介して常に呈示される。

【0886】2) 拡張/ユーザデータの最後のバイトを読み終えたことを示すイベントは無い。

【0887】A. 14. 7 「追加情報の受信」

H. 261及びMPEGは、符号化規格を拡張する情報がブロック(H. 261)又はスライス(MPEG)のピクチャ及びグループ内に埋め込まれることを可能にする。このメカニズムは、拡張データ及びユーザデータ用メカニズムと異なる(セクションA. 14. 6に記述済み)。スタートコードはデータに先行しないので、スタートコード検出器によって削除されることが可能である。

【0888】H. 261オペレーション期間中、パーザイベントERR PSPARE、及びERR GSPAREは、この情報の検出を指示する。MPEGオペレーション期間中の対応するイベントはERR EXTRAピクチャ、及びERR EXTRAスライスである。

【0889】パーザイベントが生成されると、追加情報の第1のバイトは、レジスタrom revisionを介して経て呈示される。

【0890】動画デマルチプレクサレジスタの継続状態は、イベントがクリアされた後の行動態様を決定する。このレジスタ値0を保持する場合には、あらゆる残りの追加情報は動画デマルチプレクサによって消費され、そ

して、一切のイベントが生成されない。継続が1にセットされている場合には、追加情報の各バイトが動画デマルチプレクサに到着するにつれて、イベントが生成される。この継続は、追加情報が使い果たされるか、或いは、継続が0にセットされるまで継続する。

【0891】注記：

1) 拡張/ユーザデータの第1のバイトは、継続の状態に関係なく、rom revisionレジスタを介して常に呈示され。

2) 拡張/ユーザデータの最後のバイトを読み終えたことを示すイベントは無い。

【0892】A. 14. 7. 1 「FIELD INFOトークンの生成」

MPEGオペレーション期間中、レジスタフィールドinfoが1にセットされている場合には、あらゆるextra informationピクチャの第1のバイトは、FIELD INFOトークン内に配置される。この行動態様は、MPEGの標準化アクティビティによってカバーされない。FIELD INFOトークンの定義を表4～表11に示す。

【0893】フィールドinfoが1にセットされている場合には、extra informationピクチャの第1のバイトに対して一切のパーザイベントが生成されない。ただし、extra informationピクチャのあらゆる後続バイトに対してイベントが生成される。extra informationピクチャの1つの単一バイトだけがある場合には、パーザイベントは一切発生しない。

【0894】A. 14. 8 「MPEGシーケンス層における変化」

MPEGシーケンスヘッダは、復号化されようとしている動画の次に示す特性を記述する。

【0895】*水平および垂直サイズ

*ピクセルのアスペクトレシオ

*ピクチャレート

*コード化データレート

*動画バッファベリファイアのバッファサイズ

空間デコーダがシーケンスヘッダを復号化する際に、これら任意のパラメータが変化する場合には、パーザイベントERR MPEGシーケンスが生成される。

【0896】A. 14. 8. 1 「ピクチャサイズの変化」

ピクチャサイズが変化した場合、ユーザのソフトウェアは、horizpels、及びvert垂直pelsにおける値を読み取り、そして、レジスタhorizmacroblocks、及びvertmacroblocksにロードするための新しい値を計算する。

【0897】セクション A. 15 「空間復号化」

本発明に基づき、トークンバッファの出力と空間デコーダの出力との間で空間復号化が発生する。

【0898】空間復号化の責任を負う3つの主要ユニット、即ち、逆モデラ、逆量子化器、及び逆個コサイントランスフォーマがある。このセクションの入力（トークンバッファから）において、データトークンは、量子化された係数のラン及びレベル表現を含む。出力（逆DCTの）において、データトークンはピクセル情報の8×8ブロックを含む。

【0899】A. 15. 1 「逆モデラ」

トークンバッファ内のデータトークンは、量子化された係数の値、及び表示された係数の間のゼロの数に関する情報を含む。逆モデラは、各データトークンが64の値を含むように、ゼロのランに関する情報を拡大する。この段階において、データトークンにおける値は量子化された係数である。

【0900】逆モデル化プロセスは、現在使用中の符号化規格には関係なく、同じである。コンフィギュレーションは一切不要である。

【0901】モデル化および逆モデル化機能に関する全ての必要条件を一層良く理解するために、読者は、任意のピクチャコード化規格を調査することが可能である。

【0902】A. 15. 2 「逆量子化器」

エンコーダにおいて、量子化器は、DCT係数の分解度を低下させるためにDCT出力を分割する。デコーダにおける逆量子化器の機能は、元の値に近似するように復元するために、これらの量子化されたDCT係数に乗算を施すことである。

【0903】A. 15. 2. 1 「規格量子化計画の概要」

異なる符号化規格の各々によって用いられる量子化計画には重要な差がある。各々の規格によって用いられる量子化計画について詳細に理解するためには、読者は、関連符号化規格文書を研究されたい。

【0904】レジスタiqコード化スタンダードは、異なる規格の必要条件に適合するように、逆量子化器のオペレーションを構成する。正常オペレーションにおいて、このコード化レジスタは、コーディング・スタンダードトークンによって自動的にロードされる。コード化規格のコンフィギュレーションに関する一層の情報に関しては、セクションA. 21. 1を参照されたい。

【0905】量子化計画の間の主要な相違は、量子化された係数に乗算する数のソースである。これらについては、次に概説される。更に、ここには説明されていないが、必要な算術演算（丸める、等）において詳細な相違がある。

【0906】A. 15. 2. 1. 1 「H. 261 I

Qの概観」

H. 261において、係数を調整するために、1つの単一「スケールファクタ」が用いられる。エンコーダは、作成されるデータレート进行调整するために、このスケールファクタを定期的に変更できる。イントラコード化ブロック内の「DC」係数には、わずかに異なる規則が適用される。

【0907】A. 15. 2. 1. 2 「JPEG IQの概観」

基底線JPEGは、各走査において最大4つまでの異なる色成分を含むピクチャを許容する。これらの4つの色成分に対しては、64エントリ量子化表が指定可能である。これらの表の各エントリは、64の量子化された係数のうちの1つに対して「スケール」ファクタとして使用できる。

【0908】JPEG量子化表用の値は、コード化JPEGデータ内に含まれ、そして、量子化表に自動的にロードされる。

【0909】A. 15. 2. 1. 3 「MPEG IQの概観」

MPEGは、H. 261及びJPEG量子化技術を用いる。JPEGの場合と同様に、各々64のエントリを備えた4つの量子化表が使用可能である。ただし、表の使用方法は全く異なる。

【0910】データの2つの「タイプ」は、イントラ及び非イントラである。各データタイプに対して異なった表が用いられる。2つの「デフォルト」表が、MPEGによって定義される。1つはイントラデータ用であり、いま1つは非イントラデータ用である表130、及び表131参照）。これらのデフォルト表は、MPEG復号化が可能になる以前に、空間デコーダの量子化表メモリに記入されなければならない。

【0911】同様に、MPEGは、2つの「ダウンロード」量子化表を許可する。1つは、イントラデータと共に使用するためであり、いま1つは、非イントラデータと共に使用するためである。これらの表に対する値は、MPEGデータストリーム内に含まれ、そして、量子化表メモリ自動的にロードされる。

【0912】表から出力される値は、スケールファクタによって修正される。

【0913】A. 15. 2. 2 「逆量子化器レジスタ」

【0914】

【表129】

レジスタ名	リ セ ット 状 態	記 述
iq_access	1 rw	このアクセスビットが逆量子化器の動作を停止し、その結果 個々のレジスタが高精度度を以てアクセス可能である。 A. 6. 4. 1 参照
iq_coding_standard	2 rw	このレジスタは、逆量子化器により用いられたコード化規格 を構成する。レジスタは CODING_STANDARD ト ークンにより直接ロードすることが可能である。A. 21. 1 参照
iq_keyhole_address	a rw	4つの量子化器表を保持するレジスタへのキーホールアクセ ス。キーホールを介してレジスタへのアクセスに関する更に
iq_keyhole_data	8 rw	詳細な情報については A. 6. 4. 3 参照

表 A. 15. 1 逆量子化器レジスタ

本発明において、量子化表メモリがアクセス可能になる
以前に、i q アクセスレジスタがセットされなければならない。
i q アクセスが 0 にセットされている間に、表 20
に読み取りが試行された場合には、量子化表メモリは値
ゼロを返す。

【0915】 A. 15. 2. 3 「逆量子化器の構成」
正常オペレーションにおいては、コーディング・スタン
ダードトークンによって自動的に構成されるので、逆量
子化器の符号化規格を構成する必要はない。

【0916】 H. 261 オペレーションに対しては、量
子化器表は使用されない。特別なコンフィギュレーション
は必要とされない。J PEG オペレーションに対して
は、逆量子化器によって必要とされる表は、コード化デ

ータから抽出された情報によって自動的にロードされな
ければならない。MPEG オペレーションは、デフォル
ト量子化表がロードされることを必要とする。このロー
ディングは、i q アクセスが 1 にセットされている間に
行われなければならない。表 130 内の値は、逆量子化
器の拡張アドレススペースの場所 0 x 00 から 0 x 3 F
までに記入されなければならない（キーホールレジスタ
i q キーホールアドレス、及び i q キーホールデータを
介してアクセス可能）。同様に、表 131 の値は、逆量
子化器の拡張アドレススペースの場所 0 x 40 から 0 x
7 F までに記入されなければならない。

【0917】

【表 130】

i*	W _{1,0} *	i	W _{1,0}	i	W _{1,0}	i	W _{1,0}
0	8	16	27	32	29	48	35
1	16	17	27	33	29	49	38
2	16	18	26	34	27	50	38
3	19	19	26	35	27	51	40
4	16	20	26	36	29	52	40
5	19	21	26	37	29	53	40
6	22	22	27	38	32	54	48
7	22	23	27	39	32	55	48
8	22	24	27	40	34	56	46
9	22	25	29	41	34	57	46
10	22	26	29	42	37	58	56
11	22	27	29	43	38	59	56
12	26	28	34	44	37	60	58
13	24	29	34	45	36	61	69
14	26	30	34	46	35	62	69
15	27	31	29	47	34	63	83

表 A. 15. 2 イントラコード化ブロック用デフォルト MPEG 表

a: 量子化表メモリのアダプタからのオフセット

b: 量子化表の値

【0918】

【表 131】

i	W _{i,1}	i	W _{i,1}	i	W _{i,1}	i	W _{i,1}
0	16	16	16	32	16	48	16
1	16	17	16	33	16	49	16
2	16	18	16	34	16	50	16
3	16	19	16	35	16	51	16
4	16	20	16	36	16	52	16
5	16	21	16	37	16	53	16
6	16	22	16	38	16	54	16
7	16	23	16	39	16	55	16
8	16	24	16	40	16	56	16
9	16	25	16	41	16	57	16
10	16	26	16	42	16	58	16
11	16	27	16	43	16	59	16
12	16	28	16	44	16	60	16
13	16	29	16	45	16	61	16
14	16	30	16	46	16	62	16
15	16	31	16	47	16	63	16

表A.15.3 非イントラコード化ブロック用デフォルトMPEG表

A.15.2.4 「トークンからの表構成」

MPIを介して逆量子化器表を構成する代りに、これらの逆量子化器表をトークンによって初期化することが可能である。これらのトークンは、コード化されたデータポート又はMPIのいずれかによって供給可能である。

【0919】QUANTテーブルトークンについては、表3～表11に記述される。前記トークンは、表の4（0～3）つの場所のいずれがトークンによって定義されるかを指定する1つの2ビットフィールドttを持つ。MPEGオペレーションに対しては、表0及び1のデフォルトの定義がロードされることが必要である。

【0920】A.15.2.5 「量子化表の値」

JPEG及びMPEGの両方に対しては、量子化表エンタリは、8ビットの数である。255から1までの値は

正当である。値0は不当である。

【0921】A.15.2.6 「量子化表の数順序」

量子化表の値は、「ジグザグ」走査順序において用いられる（符号化規格参照）。表は、64の値を持つ一次元アレイと見なされなければならない（8×8アレイではない）。低いアドレスにおける表のエントリは、低い周波数DCT係数に対応する。

【0922】量子化表の値がQUANTテーブルトークンによって所持される場合には、トークンヘッダの後の第1の値は、「DC」係数用の表エントリである。

【0923】A.15.2.7 「逆量子化器テストレジスタ」

【0924】

【表132】

レジスタ名	リ セ ツ ト 状 態	記 述
iq_quant_scale	5 rw	このレジスタは、量子化スケール係数の誤行値を保持し、QUANT_SCALEトークンによりロードされる。これはJ P E G 作動期間中に用いられない。
iq_component	2 rw	このレジスタは、最も最近のDATAトークンヘッドからの2ビット成分IDトークンを保持する。この値は、量子化器表の選択に關係する。更に、表をロードするためにQUANT_TABLEトークンが到着した後で表IDを保持する。
iq_prediction_mode	2 rw	これは最も最近のPREDICTION_MODEトークンの2つのLSBを保持する。
iq_jpeg_indirection	8 rw	このレジスタは、DATAトークンの2ビット成分ID番号を、使用されるべき量子化表の表番号に關係付ける。 ビット1:0 成分0と共に用いられる表番号を指定する。 ビット3:2 成分1と共に用いられる表番号を指定する。 ビット5:4 成分2と共に用いられる表番号を指定する。 ビット7:6 成分3と共に用いられる表番号を指定する。 このレジスタは、J P E G _ T A B L E _ S E L E C T トークンによりロードされる。
iq_mpeg_indirection	2 rw	この2ビットレジスタはイントラ及び非イントラデータと共にデフォルト表、又はダウンロードされた量子化表を用いるかを記録する。ビット信号0は、デフォルト表を使用すべきであることを示す。1は、ダウンロードされた表を使用すべきであることを示す。ビット0は、イントラデータに關係する。ビット1は、非イントラデータに關係する。このレジスタは通常、M P E G _ T A B L E _ S E L E C T によりロードされる。

表A. 15.4 逆量子化器テストレジスタ

A. 15. 3. 「離散逆コサイン変換」

本発明の逆別個変換プロセッサは、CCITT勧告H. 261、IEEE仕様P1180に規定された必要条件に適合し、そして、MPEGの現行改訂案に記述された必要条件に適合する。

【0925】離散逆コサイン変換プロセスは、いずれのコード化規格が用いられるかに關係なく、同じである。

ユーザによるコンフィギュレーションは一切不要である。

【0926】逆別個変換プロセッサと関連した2イベントがある。

【0927】

【表133】

レジスタ名	リ セ ツ ト 状 態	記 述
idct_too_few_event	1 rw	逆DCTは、全てのDATAトークンが正確に64個の値を含むことを要求する。値の個数が64未満の場合には過少イベントが生成される。マスクレジスタが1にセットされる場合は割込みが生成可能であり、逆DCTが停止する。このイベントはデコード化データ内のエラーに続く場合に限り生じる。
idct_too_few_mask	1 rw	
idct_too_many_event	1 rw	逆DCTは、全てのDATAトークンが正確に64個の値を含むことを要求する。値の個数が64超過の場合には過多イベントが生成される。マスクレジスタが1にセットされる場合は割込みが生成可能であり、逆DCTが停止する。このイベントはデコード化データ内のエラーに続く場合に限り生じる。
idct_too_many_mask	1 rw	

表A.15.5 逆DCTイベントレジスタ

DCT及び逆DCT機能を一層良く理解するために、読者は、任意のピクチャ符号化規格を調査することができる。

【0928】セクション A. 16 「空間デコーダ出力への接続」

空間デコーダの出力は、幅9ビットのデータワードの標準トークンポートである。インターフェースの電氣的行動態様に関する詳細な情報についてはセクションA. 4を参照されたい。

【0929】出力におけるトークンは、採用中の符号化規格に依存する。一例として、本開示のこのセクションにおいては、JPEGオペレーション用に構成する際に、空間デコーダの出力について考察する。時間デコーダは、JPEGの復号化結果として得られるトークンシーケンスを修正しないので、このセクションにおいては、JPEGオペレーション期間中に時間デコーダの出力において観察されるトークンシーケンスについて記述する。

【0930】ただし、MPEG及びH. 261は、双方共時間デコーダの使用を必要とする。MPEG及びH. 261オペレーション用に構成されている場合における時間デコーダ出力への接続に関する情報についてはセクションA. 19を参照されたい。更に、このセクションにおいては、空間デコーダの出力においていずれのトークンが利用できるか及び当該出力をディスプレイするための回路設計に際していずれのトークンが最も有用であるかを識別する。他のトークンも存在するが、それらの出力はディスプレイ不要なので、ここでは検討しない。このセクションにおいて集中的に検討する項目を次に示す。

【0931】*シーケンスのスタート及びエンドの識別方法。

【0932】*ピクチャのスタート及びエンドの識別方

法。

【0933】*何時ピクチャをディスプレイするかの識別方法。

【0934】* ディスプレイ内のピクチャデータ配置場所を識別する方法

A. 16. 1 「JPEGピクチャの構造」

このセクションでは、JPEG構文の特徴について概観する。詳細については符号化規格を参照されたい。

【0935】JPEGは、個々のピクチャをコード化するための様々なメカニズムを提供する。JPEGは、動画をコード化するメカニズムを提供するためにピクチャの集りをまとめてコード化する方法を記述しようと試みない。

【0936】本発明に基づく空間デコーダは、JPEGの基底線としての順次的な作動モードをサポートする。この構文には主要な3つのレベル、即ち、イメージ、フレーム、及びスキャンがある。順次的な1つのイメージはただ1つの単一フレームを含む。1つのフレームは、1から256までの間の異なるイメージ(色)成分を含むことができる。これらのイメージ成分は様々な方法においてスキャンに集めることができる。各スキャンは1から4までの間のイメージ成分を含むことができる(図90「JPEG基底線としての順次的構造の概観」参照)。スキャンが1つの単一イメージ成分を含む場合には、非インタリーブスキャンであり、複数のイメージ成分を含む場合には、インタリーブスキャンである。1つのフレームは、インタリーブスキャンと非インタリーブスキャンの混合物を含むことができる。フレームが含むことのできるスキャンの数は、フレームが含むことのできるイメージ成分の数に関する256の限界によって決定される。1つのインタリーブスキャン内において、データは、MPEG及びH. 261に用いられるマクロブロックに相似する最小コード化単位(M

CU)に組織される。これらのMCUは、ピクチャ内においてラスト順に配置される。非インタリーブドスキャンにおけるMCUは、1つの単一8×8ブロックである。この場合にも、MCUはラスト組織化される。

【0937】空間デコーダは、1から4までの異なる色成分を含むJPEGデータを容易に復号化することが出来る。大きい数の成分について記述するファイルも、同様に復号化することが出来る。ただし、復号化される成分の次の1組を収容するためには、スキャンの間において或る程度の再構成が要求されることもあり得る。

【0938】A. 16. 2 「トークンシーケンス」JPEGマーカコードは、スタートコード検出器によってトークンと指定された相似のMPEGに変換される表92、及び図91「トークン化されたJPEGピクチャ」参照)。

【0939】セクションA. 17 「時間デコーダ」*30、MH7オペレーション。

【0940】*MPEG及びH. 261動画デコーダ用時間復号化を提供する。

【0941】*H. 261 CIF、及びQCIFフォーマット。

【0942】*MPEG動画分解能度最大704×480、30Hz、4:2:0。

【0943】*融通性のあるクロマサンプリングフォーマット。

【0944】*MPEGピクチャシーケンス順序直し可能。

【0945】*グルーレスDRAMインターフェース。

【0946】*単一+5V電源。

【0947】*208ピンPQFPパッケージ。

【0948】*最大電力消費2.5W。

【0949】*標準ページモードDRAMを使用する。

【0950】時間デコーダは、空間デコーダにとってコンバニオンチップであり、そして、H. 261、及びMPEGによって必要とされる時間復号化を提供する。

【0951】時間デコーダは、MPEG、及びH. 261によって必要とされる全ての予測形成機能を実行する。1つの単一4Mb DRAM (例えば、512k×8)を用いて、時間デコーダは、CIF及びQCIF H. 261動画を復号化することができる。8Mb DRAM (例えば、2つの256k×16)を用いると、704×480、30Hz、4:2:0MPEG動画を復号化することができる。

【0952】イントラコード化計画 (例えばJPEG) は時間デコーダを必要としない。時間デコーダが多重規格デコーダに含まれる場合には、当該時間デコーダは、復号化済みJPEGピクチャをその出力まで供給する。

注記：前述の値は、本発明の1実施例の一例を示すに過ぎず、必ずしも制限を意図するものではない。本発明の範囲から逸脱することなしに、他の値及び範囲を用いることが可能であることが理解されるはずである。

【0953】A. 17. 1 「時間デコーダ信号」

【0954】

【表134】

信号名	I/O	ピンナンバー	記 述
in_data[8:0]	I	173, 172, 171, 169, 168, 167, 166, 164, 163	入力ポート。これは一般に空間デコードの出力ポートに接続された標準2線インタフェースである。
in_extn	I	174	A. 18.1 参照
in_valid	I	162	A. 18.1 参照
in_accept	O	161	A. 18.1 参照
enable[3:0]	I	126, 127	マイクロプロセッシングフェース (MP1)
rv	I	125	A. 8.1 参照
addr[7:0]	I	137, 136, 135, 133, 132, 131, 130, 129	A. 8.1 参照
data[7:0]	O	152, 151, 149, 147, 145, 143, 141, 140	A. 8.1 参照
irq	O	154	A. 8.1 参照
DRAM_data[31:0]	I/O	15, 17, 19, 20, 22, 25, 27, 30, 31, 33, 35, 38, 39, 42, 44, 47, 49, 57, 59, 61, 63, 66, 68, 70, 72, 74, 76, 79, 81, 83, 84, 85	DRAMインタフェース A. 5.2 参照
DRAM_addr[10:0]	O	184, 186, 188, 189, 192, 193, 195, 197, 199, 200, 203	A. 5.2 参照
EAS	O	11	A. 5.2 参照
ECS[3:0]	O	2, 4, 6, 8	A. 5.2 参照
VE	P	12	A. 5.2 参照
OE	O	204	A. 5.2 参照
DRAM_enable	I	112	A. 5.2 参照
out_data[7:0]	O	89, 90, 92, 93, 94, 95, 97, 98	出力ポート。これは標準2線インタフェースである。
out_extn	O	87	A. 4及びA. 19 参照
out_valid	O	88	A. 4及びA. 19 参照
out_accept	I	100	A. 4及びA. 19 参照
tck	I	115	JTAGポート。A. 8 参照
tdi	I	116	A. 8 参照
tdo	O	120	A. 8 参照
tms	I	117	A. 8 参照
tirst	I	121	A. 8 参照
decoder_clock	I	177	メインデコードクロック 表A. 1.2 参照
reset	I	150	リセット

表A. 17.1 時間デコード信号

【0955】

【表135】

信号名	I/O	ピンNo.	記 述
tph0ish	I	122	override=1 であれば、tph0ish 及び tph1ish は、オンチップ2相クロック用入力である。正常動作に対しては、override=0 にセットすること。tph0ish 及び tph1ish は重複される（結果としてGND又はV _{DD} に接続される）。
tph1ish	I	123	
override	I	110	正常動作に対しては、override=0 にセットすること。
chiptest	I	111	正常動作中はGND又はV _{DD} に接続すること。
tloop	I	114	正常動作中はGND又はV _{DD} に接続すること。
ramtest	I	109	ramtest=1 であれば、オンチップRAMのテストが動作化される。正常動作に対しては、ramtest=0 にセットすること。
pliselect	I	178	pliselect=0 であれば、オンチップ相固定ループは不能化される。正常動作に対しては、pliselect=1 にセットすること。
ti	I	180	テスト動作中は、DRAMインタフェースにより2つのクロックが必要とされる。正常動作中はGND又はV _{DD} に接続すること。
tq	I	179	正常動作中はGND又はV _{DD} に接続すること。
pdout	O	207	これら2つのピンは、位相クロックループに対して外部フィルタ用に接続される。
pdin	I	206	これら2つのピンは、位相クロックループに対して外部フィルタ用に接続される。

表A. 17.2 時間デコードテスト信号

【0956】

【表136】

305

306

信号名	ピン	信号名	ピン	信号名	ピン	信号名	ピン
nc	208	nc	158	nc	104	nc	53
test pin	207	nc	155	nc	103	nc	51
test pin	206	irq	154	nc	102	nc	50
GND	205	nc	153	VDD	101	DRAM_data[15]	49
OE	204	data[7]	152	out_accept	100	nc	48
DRAM_addr[0]	203	data[6]	151	out_valid	99	DRAM_data[16]	47
VDD	202	nc	150	out_data[0]	98	nc	46
nc	201	data[5]	149	out_data[1]	97	GND	45
DRAM_addr[1]	200	nc	148	GND	96	DRAM_data[17]	44
DRAM_addr[2]	199	data[4]	147	out_data[2]	95	nc	43
GND	198	GND	146	out_data[3]	94	DRAM_data[18]	42
DRAM_addr[3]	197	data[3]	145	out_data[4]	93	VDD	41
nc	196	nc	144	out_data[5]	92	nc	40

表A.17.3 時間デコードピン割当て(1/3)

[0957]

[表137]

信号名	ピン	信号名	ピン	信号名	ピン	信号名	ピン
DRAM_addr[4]	195	data[2]	143	VDD	91	DRAM_data[19]	39
VDD	184	nc	142	out_data[6]	90	DRAM_data[20]	38
DRAM_addr[5]	193	data[1]	141	out_data[7]	89	nc	37
DRAM_addr[6]	192	data[0]	140	nc	88	GND	36
nc	191	nc	139	out_extn	87	DRAM_data[21]	35
GND	190	VDD	138	GND	86	nc	34
DRAM_addr[7]	189	addr[7]	137	DRAM_data[0]	85	DRAM_data[22]	33
DRAM_addr[8]	188	addr[6]	136	DRAM_data[1]	84	VDD	32
VDD	187	addr[5]	135	DRAM_data[2]	83	DRAM_data[23]	31
DRAM_addr[9]	186	GND	134	VDD	82	DRAM_data[24]	30
nc	185	addr[4]	133	DRAM_data[3]	81	nc	29
DRAM_addr[10]	184	addr[3]	132	nc	80	GND	28
GND	183	addr[2]	131	DRAM_data[4]	79	DRAM_data[25]	27
nc	182	addr[1]	130	GND	78	nc	26
VDD	181	VDD	129	nc	77	DRAM_data[26]	25
test pin	180	addr[0]	128	DRAM_data[5]	76	nc	24
test pin	179	enable[0]	127	nc	75	VDD	23
test pin	178	enable[1]	126	DRAM_data[6]	74	DRAM_data[27]	22
decoder_clock	177	rd	125	VDD	73	nc	21
nc	176	GND	124	DRAM_data[7]	72	DRAM_data[28]	20
GND	175	test pin	123	nc	71	DRAM_data[29]	19
in_extn	174	test pin	122	DRAM_data[8]	70	GND	18
in_data[8]	173	trst	121	GND	69	DRAM_data[30]	17
in_data[7]	172	tdo	120	DRAM_data[9]	68	nc	16
in_data[6]	171	nc	119	nc	67	DRAM_data[31]	15
VDD	170	VDD	118	DRAM_data[10]	66	VDD	14
in_data[5]	169	tms	117	VDD	65	nc	13
in_data[4]	168	tdi	116	nc	64	VE	12
in_data[3]	167	tck	115	DRAM_data[11]	63	EVS	11
in_data[2]	166	test pin	114	nc	62	nc	10
GND	165	GND	113	DRAM_data[12]	61	GND	9
in_data[1]	164	DRAM_enable	112	GND	60	EVS[0]	8

表A.17.3 時間デコードピン割当て(2/3)

[0958]

50 [表138]

信号名	ピン	信号名	ピン	信号名	ピン	信号名	ピン
in_data[0]	163	test pin	111	DRAM_data[13]	59	nc	7
in_valid	162	test pin	110	nc	58	CAS[1]	6
in_accept	161	test pin	109	DRAM_data[14]	57	VDD	5
reset	160	nc	108	VDD	56	CAS[2]	4
VDD	159	nc	107	nc	55	nc	3
nc	158	nc	106	nc	54	CAS[3]	2
nc	157	nc	105	nc	53	nc	1

表A.17.3 時間デコーダピン割当て(3/3)

A.17.1.1 「非接続ピン“nc”」

表138に「nc」とラベル表示されているピンは、現在本発明には使用されず、将来の製品のために予約されているピンである。これらのピンは、接続しないで残しておかねばならない。これらのピンは、VDD、GND、相互、または他のあらゆる信号と接続してはならない。

【0959】A.17.1.2 「VDD、及びGNDピン」

当然理解されるように、全てのVDD及びGNDピンは該

当する電源に接続されなければならない。全てのVDD及びGNDピンが正しく使用されていなければ、デバイスは正しく作動しない。

【0960】A.17.1.3 「正常作動のためのテストピンの接続」

時間デコーダの9つのピンは、内部テスト用として予約済みである。

【0961】

【表139】

ピンナンバー	接 続
	正常作動に対しては、GNDに接続すること。
	正常作動に対しては、VDDに接続すること。
	正常作動に対しては、開回路のまゝにしておくこと。

表A.17.4 デフォルトテストピンの接続

A.17.1.4 「正常作動のためのJTAGピン」 30 【0962】

セクションA.8.1参照。

【表140】

アドレス[16進]	レジスタ名	表参照
0x00...0x01	部込みサービスエリア	A.17.6
0x02...0x07	使用せず	
0x08	チップアクセス	A.17.7
0x09...0x0F	使用せず	
0x10	ピクチャシーケンス	A.17.8
0x11...0x1F	使用せず	
0x20...0x2E	DRAMインタフェース構成レジスタ	A.17.9
0x2F...0x3F	使用せず	
0x40...0x53	バッファ構成	A.17.8
0x54...0x5F	使用せず	
0x60...0xFF	テストレジスタ	A.17.11

表A.17.5 時間デコーダメモリマップの概観

【0963】

【表141】

アドレス (16進)	ビット 番号	レジスタ名	参照頁
0x00	7	chip_event	
	6:2	使用せず	
	1	chip_stopped_event	
	0	count_error_event	
0x01	7	chip_mask	
	6:2	使用せず	
	1	chip_stopped_mask	
	0	count_error_mask	

表A.17.6 割込みサービス領域レジスタ

【0964】

【表142】

アドレス (16進)	ビット 番号	レジスタ名	参照頁
0x08	7:1	使用せず	
	0	chip_access	

表A.17.7 チップアクセスレジスタ

【0965】

【表143】

アドレス (16進)	ビット 番号	レジスタ名	参照頁
0x10	7:1	使用せず	
	0	MPEG_reordering	

表A.17.8 ピクチャシーケンシング

【0966】

【表144】

アドレス (16進)	ビット 番号	レジスタ名	参照頁
0x20	7:5	使用せず	
	4:0	page_start_length[4:0]	
0x21	7:4	使用せず	
	3:0	read_cycle_length[3:0]	
0x22	7:4	使用せず	
	3:0	write_cycle_length[3:0]	
0x23	7:4	使用せず	
	3:0	refresh_cycle_length[3:0]	
0x24	7:4	使用せず	
	3:0	CAS_falling[3:0]	
0x25	7:4	使用せず	
	3:0	RAS_falling[3:0]	
0x26	7:1	使用せず	
	0	interface_tuning_access	
0x27	7:0	使用せず	
0x28	7:6	RAS_strength[2:0]	
	5:3	OEVE_strength[3:0]	
	2:0	DRAM_data_strength[3:0]	
0x29	7	使用せず	
	6:4	DRAM_addr_strength[3:0]	
	3:1	CAS_strength[3:0]	
	0	RAS_strength[3]	

表A.17.9 DRAMインタフェースコンフィギュレーションレジスタ

【0967】

【表145】

アドレス (16進)	ビット 番号	レジスタ名	参照頁
0x28	7	使用せず	
	6:4	DRAM_addr_strength[3:0]	
	3:1	CAS_strength[3:0]	
	0	RAS_strength[3]	
0x29	7:6	RAS_strength[2:0]	
	5:3	OEVE_strength[3:0]	
	2:0	DRAM_data_strength[3:0]	
0x2A	7:0	refresh_interval	
0x2B	7:0	使用せず	
0x2C	7:6	使用せず	
	5	DRAM_enable	
	4	no_refresh	
	3:2	row_address_bits[1:0]	
	1:0	DRAM_data_width[1:0]	
0x2D	7:0	使用せず	
0x2E	7:0	テストレジスタ	

表A.17.9 DRAMインタフェースコンフィギュレーションレジスタ

【0968】

【表146】

アドレス (16進)	ビット 番号	レジスタ名	参照頁
0x40	7:0	使用せず	
0x41	7:2		
	1:0	picture_buffer_0(17:0)	
0x42	7:0		
0x43	7:0		
0x44	7:0	使用せず	
0x45	7:2		
	1:0	picture_buffer_1(17:0)	
0x46	7:0		
0x47	7:0		

表A.17.10 バッファコンフィギュレーションレジスタ(1/2)

【0969】

【表147】

アドレス (16進)	ビット 番号	レジスタ名	参照頁
0x48	7:0	使用せず	
0x49	7:1		
	0	component_offset_0(16:0)	
0x4A	7:0		
0x4B	7:0		
0x4C	7:0	使用せず	
0x4D	7:1		
	0	component_offset_1(16:0)	
0x4E	7:0		
0x4F	7:0		
0x50	7:0	使用せず	
0x51	7:1		
	0	component_offset_2(16:0)	
0x52	7:0		
0x53	7:0		

表A.17.10 バッファコンフィギュレーションレジスタ(2/2)

【0970】

【表148】

アドレス (16進)	ビット 番号	レジスタ名	参照頁
0x22	7...4	PIL_resistors	
	3...0		
0x60	7...6	使用せず	
	5...4	coding_standard[1:0]	
	3...2	picture_type[1:0]	
	1	R_261_filt	
	0	R_261_sf	
0x61	7...6	component_id	
	5...4	prediction_mode	
	3...0	sax_sampling	
0x62	7...0	samp_h	
0x63	7...0	samp_v	
0x64	7...0	back_h	
0x65	7...0		
0x66	7...0	back_v	
0x67	7...0		
0x68	7...0	forw_h	
0x69	7...0		
0x6A	7...0	forw_v	
0x6B	7...0		
0x6C	7...0	width_in_sq	
0x6D	7...0		

表A.17.11 テストレジスタ

セクションA. 18 「時間デコーダオペレーション」

A. 18. 1 「データ入力」

時間デコーダの入力データポートは9ビット幅のデータワードを持つ標準のトークンポートである。ほとんどの応用例において、これは空間デコーダの出力トークンポートに直接接続される。このインターフェースの電気的行為に関する詳細な情報については、セクションA. 4を参照せよ。

【0971】 A. 18. 2 「自動形成」

符号化されたビデオピクチャフォーマットに関するパラメータは、空間デコーダによって発生されるトークンにより、時間デコーダ内のレジスタに自動的にロードされる。

【0972】

【表149】

トークン	実施される構成
CODING STANDARD	時間デコーダのコーディングスタンダードは、CODINGSTANDARDトークンによって自動的に形成される。これは新しいシーケンスがスタートされる度に空間デコーダによって発生される。図A. 21. 1を参照。
DEFINE SAMPLING	各々の色成分用の水平及び垂直のクロマサンプリング情報はDEFINESAMPLINGトークンによって自動的に形成される。
HORIZONTAL MAS	マクロブロック内のピクチャの水平幅はHORIZONTALMASトークンにより自動的に形成される。

表A. 18. 1 トークンを介しての時間デコーダの構成

A. 18. 3 「手動形成」

ユーザーは（マイクロプロセッサインターフェースを介

して）アプリケーション依存要素を形成しなければなら

ない。

【0973】A. 18. 3. 1 「いつ形成するか」
データ処理が行われていない時のみ、時間デコードを形成すべきである。これはリセットが解除された後のデフォルト状態である。チップアクセスレジスタに1を書き込むことにより、時間デコードを停止して再形成できる。形成が完了した後、チップアクセスに0を書き込まねばならない。

【0974】DRAMインターフェースをいつ形成するかに関する詳細は、セクションA. 5. 3を参照せよ。

【0975】A. 18. 3. 2 「DRAMインターフェース」

DRAMインターフェースのタイミングは予測的に符号化されたビデオ（例えばH. 162またはMPEG）を解読できるようになる前に形成されなければならない。セクションA. 5 「DRAMインターフェース」を参照せよ。

【0976】

【表150】

レジスタ名	書き込み 方向	リセット	説明
chip access	1 rw	1	chip accessに1を書き込むと、時間デコードがオペレーションを停止し、最速形成を可能にするように要求する。時間デコードは、通常現在のビデオシーケンスの終わりに達するまでオペレーションを続ける。リセットが解除された後、chip access=1、つまり時間デコードが停止される。
chip stopped event	1 rw	0	チップが停止すると、チップ停止イベントが発生する。chip stop event mask=1であれば、割り込みが発生する。
chip stopped mask	1 rw	0	時間デコードは誤差データに予想を加えるアダーを有する。誤差データバイト数と予想データバイト数の間に差があればカウント誤差イベントが発生する。count error mask=1であれば、割り込みが発生し、予想形成が停止する。このイベントはハードウェア誤差に続いて発生するだけである。
count error event	1 rw	0	
count error mask	1 rw	0	

表A. 18. 2 時間デコードレジスタ (1/2)

【0977】

30 【表151】

レジスタ名	サイズ 単位	ビット	説明
picture buffer 0	18 rw	x	これらはピクチャバッファ用のベース アドレスを明記する。
picture buffer 1	18 rw	x	
component offset 0	17 rw	x	これらは各々の色成分が保管される ピクチャバッファポインタからのオフ セットを明記する。component offset=nにより指示される位 置で始まる。成分ID=nを持つデータ が保管される。A. 3. 5. 1. 「成分 特定数」を参照せよ。
component offset 1	17 rw	x	
component offset 2	17 rw	x	
MPEG recording	1 rw	0	このレジスタを1に設定すると、時間デ コードは非因果関係MPEGピクチャ シーケンスからのピクチャ順序を、正 確な表示順序に配置する。 A. 18. 3. 5を参照せよ。 このレジスタはJPEG及びH. 261 のオペレーション中は無視される。

表A. 18. 2 時間デコードレジスタ (2/2)

A. 18. 3. 3 「ピクチャバッファレジスタ数」
ピクチャバッファポインタ (18ビット) 及び成分オフ
セット (17ビット) レジスタはバイトアドレスではな
く、ブロック (8×8バイト) アドレスを明記する。

【0978】A. 18. 3. 4 「ピクチャバッファ記
憶割当」

予測的に符号化されたビデオ (H. 261もしくはMP
EG) を解読するため、時間デコードは2つのピクチャ
バッファを処理しなければならない。これらのバッファ
が如何に使用されるかに関する詳細については、セクシ
ョンA. 18. 4及びA. 18. 4. 4を参照せよ。

【0979】ユーザーは (他のピクチャバッファと重ね
ることなく) 必要なビデオフォーマットの単一ピクチャ
を記憶するために、ピクチャバッファポインタ (pic
ture buffer 0及びpicture bu
ffer 1) の各々の上に十分な記憶があることを確
かめねばならない。通常、ピクチャバッファポインタの
1つが0 (つまり記憶の底部) に設定され、他の1つが
記憶スペースの中間を指すように設定される。

【0980】A. 18. 3. 4. 1 「MPEGもしく
はH. 261用の通常形成」

H. 261及びMPEGは両者共、異なる色成分間の比
4:1:1 (つまり、いずれかのクロミナンス成分にあ
るピクセルの4倍の輝度ピクセルがある) を使用する。

【0981】A. 3. 5. 1. 「成分特定数」に文書化
されているように、成分0は輝度成分であり、成分1及
び2はクロミナンスである。

【0982】成分オフセットレジスタの形成例は、成分
0がピクチャバッファポインタにおいて始まるように、
component offset 0を0に設定す
る。同様に、component offset 1は
ピクチャバッファサイズの4/6に設定し、compo
nent offset 2はピクチャバッファサイズの
5/6に設定することができよう。

【0983】A. 18. 3. 5 「ピクチャシーケンス
再整理 (re-ordering)」

MPEGは3つの異なるピクチャタイプを使用する：イ
ントラ (I)、予想された (P)、及び二方向的に補間
された (B)。Bピクチャは2つのピクチャ：未来から
のものと過去からのものからの予想に基づいている。ピ
クチャの順序は、Bピクチャの解読が求められる前に、
I及びPピクチャが符号化されたデータから解読される
ように、エンコーダで修正される。

【0984】ピクチャシーケンスはこれらのピクチャが
表示される前に修正されなければならない。時間デコー
ダは (レジスタMPEG reordering=1に
設定することにより) このピクチャ再整理を提供す
ることができる。あるいは、ユーザーは彼の表示インタ
ーフェイス機能の一部としてピクチャ再整理を実行した
いと望むかもしれない。ピクチャ再整理を提供するよう
に時間デコードを形成することにより、解読され得るビ
デオ解像度を低減させるかもしれない。A. 18. 5を
参照せよ。

50 【0985】A. 18. 4 「予測形成」

H. 261デコーディング及びMPEGデコーディングの予測形成要件は全く異なっている。コーディングスタンダードトークンは時間デコーダが異なる基準の予測要件を収容するように自動的に形成する。

【0986】A. 18. 4. 1 「JPEGオペレーション」

JPEGオペレーション用に形成されると、JPEGは時間デコーディングを必要としないので、予測は行われない。

【0987】A. 18. 4. 2 「H. 261オペレーション」

H. 261では、予測はデコーディングされたピクチャからのみ行われる。動きベクトルは整数ピクセルの精度に条件として指定されるだけである。エンコーダは低域フィルタが予測結果に適用されるように指定することができる。

【0988】各ピクチャがデコーディングされるにつれて、それは次のピクチャをデコーディングする際に使用できるように、オフチップDRAM内のピクチャバッファに書き込まれる。デコーディングされたピクチャはオフチップDRAMに書き込まれるにつれて、時間デコーダの出力に現れる。

【0989】予測に関する詳細及び関係する演算オペレーションについては、読出し装置はH. 261スタンダードに向けられる。本発明の時間デコーダはH. 261の要件を完全に遵守している。

【0990】A. 18. 4. 3 MPEGオペレーション（再整理を伴わない）

時間デコーダのオペレーションは3つの異なるMPEGピクチャタイプ（I、P、B）の各々に対して変化する。Iピクチャは時間デコーダによる更なるデコーディングを必要としないが、後にP及びBピクチャをデコーディングする際に使用されるためにピクチャバッファ（フレーム記憶装置）に記憶されなければならない。

【0991】Pピクチャのデコーディングには、以前にデコーディングされたPもしくはIピクチャからの予測を形成する必要がある。解読されたPピクチャはP及びBピクチャを解読する際に使用するため、ピクチャバッファの中に記憶される。MPEGは動きベクトルが半ピクセルの精度に指定されるようにする。オンチップフィルタはこの半ピクセル精度を支持するための補間を提供する。

【0992】Bピクチャは両方のピクチャバッファからの予測を必要とすることができる。Pピクチャと同様に、半ピクセル動きベクトル解像度精度はピクチャ情報のオンチップ補間を要求する。Bピクチャはオフチップバッファには記憶されない。それらは一時的なものにすぎない。

【0993】全てのピクチャはそれらの符号が解読されるにつれて、時間デコーダの出力ポートに現れる。従っ

て、ピクチャシーケンスは符号化されたMPEGデータにおけるものと同じである（図94の上位部分参照）。

【0994】予測に関する詳細及び関係する演算オペレーションについては、提案されているMPEGスタンダードの草案を参照せよ。本発明の時間デコーダによりこれらの要件が満たされる。

【0995】A. 18. 4. 4 「MPEGオペレーション（再整理を伴う）」

ピクチャ再整理を伴うMPEGオペレーション（MPEG reordering=1）のために形成されると、予測形成オペレーションは上述のセクションA. 18. 4. 3におけるのと同様である。しかしながら、ピクチャシーケンスを再整理するために付加的なデータ伝送が実施される。

【0996】BピクチャデコーディングはセクションA. 18. 4. 3において記した通りである。しかしながら、I及びPピクチャはデコーディングされた時に出力されない。その代わりに、（前述のように）オフチップバッファに書き込まれ、次のIもしくはPピクチャがデコーディングのために届いた時にのみ読み出される。

A. 18. 4. 4. 1 「デコーダスタートアップ特性」

次のP（もしくはI）ピクチャのデコーディングがスタートされるまで、最初のIピクチャの出力が遅らされる。このことはビデオデコーダのスタートアップ特性を概算する時に考慮されなければならない。

【0997】A. 18. 4. 4. 2 「デコーダ停止特性」

時間デコーダは前のピクチャをそのオフチップバッファ（フレーム記憶装置）からフラッシュするために、次のPもしくはIピクチャに依存する。これはビデオシーケンスの終了時、及び新しいビデオシーケンスをスタートする時に結果が現れる。空間デコーダは最後のP（もしくはI）ピクチャをフラッシュするために、ビデオシーケンスの終了時に「偽の」I/Pピクチャを作り出すための便宜を提供する。しかしながら、この「偽の」ピクチャは次のビデオシーケンスが始まるとフラッシュされる。

【0998】空間デコーダはこの「偽の」ピクチャを抑制するためのオプションを提供する。これは、新ビデオシーケンスが旧シーケンスの完了後直ちにデコーダに供給されることが知られている場合に有用である。この新シーケンス内の最初のピクチャは前のシーケンスの最後のピクチャをフラッシュすることになる。

【0999】A. 18. 5 「ビデオ解像度」
MPEGをデコーディングする時に、時間デコーダが支持することができるビデオ解像度は、そのDRAMインターフェースのメモリー帯域幅により制限される。MPEGにとって、2つの場合を考慮する必要がある：MP

EGピクチャ再整理を伴う場合と、伴わない場合である。

【1000】セクションA. 18. 5. 2及びA. 18. 5. 3はMPEG規約の現行の草案により要求される最悪の場合の要件について論じている。より低いメモリー帯域幅の要件を持つMPEGのサブセットが企図される。例えば、整数解像度動きベクトルだけを用いて、もしくはその代わりに、Bピクチャを使用せずに、メモリー帯域幅の要件を重大に低減させる。該かるサブセットについての分析はここでは行わない。

【1001】A. 18. 5. 1 [DRAMインターフェースの特性]

DRAMインターフェースを横切ってデータを伝送するために取られるサイクル数は要因数に依存する：

・使用されるDRAMに適合させるためのDRAMイン

ターフェースのタイミング形成

・データバス幅（8、16または32ビット）、データ伝送タイプ：

- ・8×8ブロックの読出しまたは書込み
- ・半ピクセル精度に対する予測のため
- ・整数ピクセル精度に対する予測のため

DRAMインターフェースの詳細な形成に関する情報に関しては、セクションA. 5、「DRAMインターフェース」を参照せよ。

10 【1002】表152はデータ伝送の各タイプのために幾つのDRAMインターフェース「サイクル」が必要であるかを示している。

【1003】

【表152】

データバス幅 (ビット)	8×8ブロック の読出し または書込み	形態予想 (半ピクセル精度)	形態予想 (整数 ピクセル精度)
8	1ページアドレス +64伝送	4ページアドレス +81伝送	4ページアドレス +64伝送
16	1ページアドレス +32伝送	4ページアドレス +45伝送	4ページアドレス +40伝送
32	1ページアドレス +16伝送	4ページアドレス +27伝送	4ページアドレス +24伝送

表A. 18. 3 時間デコーダに対するデータ伝送回数

表154は表152の数字を取り、それらを「典型的な」DRAMのために評価する。この例では、27MHzクロックであると仮定する。ここでは27MHzを使用するが、それに制限するものではないことを認識すべきである。アクセススタートには11ティック（102ns）が必要であり、データ伝送には6ティック（56ns）が必要である。

【1004】A. 18. 5. 2 「再整理を伴わないM

PEG解像度」

ピークメモリー帯域幅のロードはBピクチャをデコーディングする時に発生する。「最悪の場合の」シナリオでは、Bフレームは両方のピクチャバッファからの予測から形成され、全ての予測は半ピクセル精度に対するものである。

【1005】

【表153】

データバス幅 (ビット)	8×8ブロック の読出し または書込み	形態予想 (半ピクセル精度)	形態予想 (整数 ピクセル精度)
8	3657ns	4907ns	3963ns
16	1880ns	2907ns	2185ns
32	991ns	1907ns	1741ns

表A. 18. 4 「典型的な」DRAMでの描写

表153からの数字例を用いれば、（32ビット幅のインターフェースを介して）2つの正確な半ピクセル精度予測のために必要なデータを読むために、DRAMインターフェース3815nsが必要であることが解る。時間デコーダが支持することができる解像度は、1ピクチャ時間内に遂行できるこれらの予測数によって決定され

る。本例では、時間デコーダは1つの33msピクチャ周期に（例えば、30Hzビデオのために）8737の8×8ブロックを処理することができる。

【1006】必要なビデオフォーマットが704×480であれば、（4：2：0のクロマサンプリングを考慮すれば）各ピクチャは7920の8×8ブロックを包含

する、このビデオフォーマットは(DRAMリフレッシュ等の他の要因を考慮に入れる前に)利用できるDRAMインターフェース帯域幅の約91%を消費する。従って、時間デコーダはこのビデオフォーマットを支持することができる。

【1007】A. 18. 5. 3 「再整理を伴うMPEG解像度」

MPEGピクチャ再整理を用いれば、Pピクチャがデコーディングされている間に、最悪の場合のシナリオと遭遇する。この時間の間に、DRAMインターフェース上 10
に3回のロードが行われる:

- ・形態予測
- ・結果を書き戻す
- ・以前のPまたはIピクチャを読み出す。

【1008】表152からの数字例を用いて、32ビット幅のインターフェースが利用できる場合に、これら各々の仕事のために必要な回数を見出すことができる。予測形成には1907ns/nが必要である一方、読出し及び書き込みには各々991nsが必要であり、全体で3889nsを必要とする。これは時間デコーダが33 20
ms周期の間に8485の8×8ブロックを処理できるようにする。

【1009】従って、704×480ビデオを処理するには、(リフレッシュを無視して)利用できるメモリ帯域幅のほぼ93%を使用することになる。

【1010】A. 18. 5. 4 「H. 261」

H. 261は30Hzまでのピクチャ率でCIF(352×288)とQCIF(172×144)の2つのピクチャフォーマットを支持するだけである。CIFピクチャは2376の8×8ブロックを包含する。唯一必要 30
なメモリーオペレーションは8×8ブロックの配線と、整数精度動きベクトルでの予測形成である。

【1011】8ビット幅のメモリーインターフェースのための表153からの数字例を用いて、各ブロックに書き込むために3657nsを必要とする一方、1ブロックの予測形成には3963ns/nが必要であり、全体でブロック毎に7620nsが必要であることが解る。1つのCIFピクチャのための処理時間は約18msであり、30Hzのビデオを支持するために必要な33msよりかなり少なくなっている。

【1012】A. 18. 5. 5 「JPEG」

支持できるJPEGビデオの解像度は発明の空間デコーダもしくは表示インターフェースの能力により決定される。時間デコーダはJPEG解像度に影響を及ぼさない。

【1013】A. 18. 6 「イベント及びエラー」

A. 18. 6. 1 「チップストップ」

本発明では、チップアクセスに1を書き込むと、時間デコーダがオペレーションを停止して再形成ができるように 50
することを求める。一度受け入れられると、通常時間

デコーダは現在のビデオシーケンスのエンドに達するまでオペレーションを続ける。その後、時間デコーダは停止される。

【1014】チップが停止すると、チップ停止イベントが発生する。chip stopped mask=1であれば、割り込みが発生する。

【1015】A. 18. 6. 2 「カウントエラー」

本発明の時間デコーダは誤差データに対する予測を加えるアダーを具備する。誤差データバイト数と予測データバイト数との間に差がある場合、カウントエラーイベントが発生する。

【1016】count error mask=1の場合、割り込みが発生し、予測形成が停止する。

【1017】count error eventに1を書き込むと、イベントをクリアして時間デコーダが進めるようにする。その後エラーを発生させたDATAトークンが続く。しかしながら、エラーを発生させたDATAトークンは正しい長さ(46バイト)のものではないであろう。これは更なる問題を容易に発生させるであろう。このように、カウントエラーは重大なハードウェアエラーが発生した場合にのみ生じるべきである。

【1018】セクションA. 19 「時間デコーダの出力への接続」

時間デコーダの出力は8ビット幅のデータワードを持つ標準のトークンポートである。インターフェースの電気的行為に関するより詳細な情報についてはセクションA. 4を参照せよ。

【1019】時間デコーダの出力に存在するトークンは使用されるコーディングスタンダードに依存し、MPEGの場合には、ピクチャが再整理されるか否かによるであろう。本セクションは時間デコーダの出力においてどのトークンが利用できるか、またその出力を表示するための回路を設計する際にどれが最も有用であるかを特定する。他のトークンも存在するであろうが、出力を表示する必要がないので、それらに関してはここでは論じない。

【1020】本セクションは以下の点を重点的に取り上げて論じる:

- ・シーケンスのスタート及びエンドを如何にして特定できるか、
- ・ピクチャのスタート及びエンドを如何にして特定できるか、
- ・ピクチャをいつ表示するかを如何にして特定するか、
- ・ピクチャデータを表示のどこに置くべきかを如何にして特定するか。

【1021】A. 19. 1 「JPEG出力」

JPEGデータのデコーディング時に、時間デコーダにより出力されるトークンシーケンスは、空間デコーダの出力において見られるものと同一である。JPEGは時間デコーダによる処理を必要としないことを思いだして

ほしい。しかしながら、時間デコーダは(空間デコーダにおけるIPCTの限定された算術精度から生じる)負のバリュウのために、イントラデータトークンを調べ、それらを0に置き換える。

【1022】JPEGオペレーション中に観察される出力シーケンスについての詳細な議論はセクションA. 16を参照せよ。

【1023】A. 19. 2 「H. 261出力」

A. 19. 2. 1 「セッションのスタートとエンド」

H. 261はビデオデータ内のビデオストリームのスタート及びエンドを合図しない。それにもかかわらず、これはアプリケーションによって暗示される。例えば、電気通信が接続されるとシーケンスが始まり、ラインが遮断されると終了する。このように、ビデオシンタックスにおいて最も高いレイヤは「ピクチャレイヤ」である。

【1024】本発明による空間デコーダのスタートコード検出器は、最初のピクチャスタートの前に、シーケンススタートとコーディングスタンダードトークンが自動的に挿入されるようにする。セクションA. 11. 7. 3及びA. 11. 7. 4を参照せよ。

【1025】H. 261セッションの終わりに(例えば、ラインが遮断された時に)、ユーザーは符号化データの終了後、フラッシュトークンを挿入しなければならない。これには多くの効果がある(A. 31. 1を参照)：

・それは最後のピクチャの終わりを合図するために、ピクチャエンドが発せられることを保証する。

【1026】・それは符号化データの終わりがデコーダを通して押されることを保証する。

A. 19. 2. 2 「ピクチャの取得」

各ピクチャはシンタックスレイヤと称されるエレメントの階層から成る。H. 261をデコーディングする時、時間デコーダの出力におけるトークンのシーケンスはこの構造を反映する。

【1027】A. 19. 2. 1 「ピクチャレイヤ」

各ピクチャの前にピクチャスタートトークンが置かれ、ピクチャのすぐ後にピクチャエンドトークンが続く。

H. 261は当然ピクチャエンドを含まない。このトークンは空間デコーダのスタートコード検出器により自動的に挿入される。ピクチャスタートトークンの後に、時間標準トークン及びピクチャタイプトークンが続くであろう。時間標準トークンは、ピクチャがいつ表示されるべきかを指示する10ビット数を支持し(その内5LSBだけがH. 261において使用される)。H. 261エンコーダは(低いデータ率を達成するために)シーケンスからピクチャを省略することができるので、この点は表示システムによって研究されるべきである。ピクチャの省略は、連続するピクチャ間の1つ以上の分だけ増加する時間基準によって検出され得る。

【1028】次に、ピクチャタイプトークンはピクチャ

フォーマットに関する情報を有する。表示システムはこの情報を調べ、CIFもしくはQCIFピクチャがデコーディングされているか否かを検出することができる。

しかしながら、ハフマンデコーダ内のレジスタを調べることで、ピクチャフォーマットに関する情報も利用できる。

<Xref:ハフマンデコーダセクション>

A. 19. 2. 2. 2 「ブロックレイヤのグループ」

各H. 261ピクチャは多数の「ブロックグループ」により構成される。その各々の前に、(H. 261のグループNo. とグループスタートコードから引き出された)スライススタートトークンが置かれる。このトークンは表示のどこにブロックのグループを置くべきかを指示する8ビット値を所有している。これはデータエラーの後デコーダが再び同時性を持つ機会を提供する。更に、それは、ピクチャを描写するために付加的な情報を必要としないピクチャエリアがあれば、ブロックを飛び越す機構をエンコーダに提供する。空間デコーダ及び時間デコーダは各ピクチャが正しい数のブロックを含み、それらのブロックが正しい位置にあることを確実にするため、既にこの情報を使用しているので、スライススタートが時間デコーダの出力に達する時までに、その情報は効果的に冗長となっている。こうして、ピクチャのスタート以来出力されてきたブロック数を数えることにより、時間デコーダにより出力されるデータブロックをどこに置くべきかを計算することが可能となる。

【1029】スライススタートにより支持される数は、H. 261のブロック数のグループより1つ少ない数である(詳細な情報についてはH. 261基準を参照せよ)。図103はCIF及びQCIFピクチャ内のH. 261ブロックグループの位置付けを示している。

【1030】注：本発明においては、図示したブロックナンバリングはスライススタートにより支持されるものと同じである。これはこれらのグループをナンバリングするためのH. 261規定とは異なる。

【1031】(各ブロックグループのスタートを示す)スライススタートと最初のマクロブロックとの間に、他のトークンがあってもよい。それらはピクチャデータを表示する必要がないので、無視できる。

【1032】A. 19. 2. 2. 3 「マクロブロックレイヤ」

ブロックの各グループ内のマクロブロックのシーケンスはH. 261により限定される。各マクロブロックの位置を説明する特別なトークン情報はない。ユーザーは各情報をどこに表示するかを決定するためにマクロブロックシーケンスを通して数えなければならない。

【1033】図105はマクロブロックがブロックの各グループに配置されるシーケンスを示している。

【1034】各マクロブロックは6個のデータトークンを包含する。6個の各グループに含まれるデータトーク

ンのシーケンスはH. 261のマクロブロック構造により限定される。各データトークンは1つの色成分の8×8ピクセルエリアのために正確に64データバイトを含むべきである。色成分はデータトークン内の2ビットの数の中に含まれる(セクションA. 3. 5. 1参照)。しかしながら、H. 261内の色成分のシーケンスは限定される。

【1035】データトークンの各グループの前には、動きベクトル、量子化器スケール係数等に関する情報を伝達する多くのトークンが置かれる。これらのトークンはピクチャが表示されるようにする必要がないので、無視することができる。

【1036】各データトークンは8×8の1つの色成分のために64データバイトを包含する。これらはラスタ状態である。

【1037】A. 19. 3 「MPEG出力」
MPEGはそのシンタックスの中に多くのレイヤを含む。これらはビデオシーケンスやピクチャグループ等の概念を具体化する。

【1038】A. 19. 3. 1 「MPEGシーケンスレイヤ」

シーケンスは多重エントリポイント(シーケンススタート)を持つことができるが、1つだけの出口ポイント(シーケンス・エンド)を持つべきである。MPEGシーケンスヘッダコードが解読される時、空間デコーダはコーディングスタンダードトークン及びそれに続くシーケンススタートトークンを生じさせる。

【1039】シーケンススタートの後、ビデオフォーマット等を記述するシーケンスヘッダ情報の多くのトークンがあるであろう。シーケンスヘッダにおいて合図される情報については草案のMPEGスタンダードを参照し、このデータが如何にしてトークンに変換されるかに関する情報については表3～表11を参照せよ。ビデオフォーマットを記述するこの情報は、ハフマンデコーダ内のレジスタにおいても利用できる。

【1040】このシーケンスヘッダ情報はMPEGシーケンス内において、もしそのシーケンスが数個のエントリポイントを持っていれば、数回発生するかもしれない。

【1041】A. 19. 3. 2 「ピクチャレイヤグループ」

ピクチャのMPEGグループはシーケンススタート時点で提供されるものに、異なるタイプの「エントリ」ポイントを提供する。シーケンスヘッダはピクチャ/ビデオフォーマットに関する情報を提供する。従って、デコーダがシーケンスにおいて使用されるビデオフォーマットの知識を持たない場合、シーケンススタートで開始しなければならない。しかしながら、一度ビデオフォーマットがデコーダの中に構成されると、どのグループのピクチャの位置であってもデコーディングを開始することが

できるようにするべきである。

【1042】MPEGはグループ内のピクチャ数を制限しない。しかしながら、多くのアプリケーションにおいて、1つのグループはランダムアクセスの道理的な粒度を提供するので、約0. 5秒に相当する。

【1043】ピクチャグループのスタートはグループスタートトークンにより指示される。グループスタートの後で提供されるヘッダ情報は2つの有益なトークン: TIME CODE及びBROKEN CLOSEDを含む。

【1044】TIME CODEはSMPTEタイムコード情報のサブセットを所有する。これはビデオデコーダを他の信号に同期させる際に有用であるかもしれない。BROKEN CLOSEDはMPEGのclosed gapとbroken linkビットを所有する。ランダムアクセスの含意及び編集されたビデオシーケンスのデコーディングに関しては、セクションA. 19. 3. 8を参照せよ。

A. 19. 3. 3 ピクチャレイヤ

新しいピクチャのスタートはピクチャスタートトークンにより指示される。このトークンの後には、時間標準トークン及びピクチャタイプトークンがある。時間デコーダがピクチャ再整理を提供するように構成されない場合、一時的な基準情報が有用であるかもしれない。ピクチャタイプの情報は、表示システムが特にオープンGOPのスタートでBピクチャを処理したい場合に有用であるかもしれない(セクション A. 19. 3. 8参照)。

【1045】各ピクチャは多くのスライスで構成される。

【1046】A. 19. 3. 4 「スライスレイヤ」
セクションA. 19. 2. 2. 2はH. 261において使用されるブロックグループについて論じている。MPEGにおけるスライスは同様の機能を果たす。しかしながら、スライス構造は基準によって固定されない。スライススタートトークンが所有する8ビット値は、MPEGが伝達する「スライス垂直位置」より1つ少ない値である。スライスレイヤの説明に関しては、MPEGスタンダード草案を参照せよ。

【1047】空間デコーダ及び時間デコーダは各ピクチャが正しい数のブロックを正しい位置に含んでいることを確実にするため、既にこの情報を使用しているので、スライススタートが時間デコーダの出力に達する時まで、その情報は効果的に冗長となっている。こうして、ピクチャのスタート以来出力されてきたブロック数を数えることにより、時間デコーダにより出力されるデータブロックをどこに置くべきかを計算することが可能となる。

【1048】MPEGピクチャ再整理を使用する効果に関する議論については、セクションA. 19. 3. 7を

参照せよ。

【1049】A. 19. 3. 5 「マクロブロックレイヤ」

各マクロブロックは6ブロックを含む。これらは(MPEG規約の草案により明記されるように)ラスト状態で時間デコーダの出力に現れる。

【1050】A. 19. 3. 6 「ブロックレイヤ」

各マクロブロックは6個のデータトークンを包含する。6個の各グループに含まれるデータトークンのシーケンスはMPEG規約草案(これはH. 261のマクロブロック構造と同じである)により限定される。各データトークンは1つの色成分の8×8ピクセルエリアのために正確に64データバイトを含むべきである。色成分はデータトークン内の2ビットの数の中に含まれる(A. 3. 5. 1参照)。しかしながら、MPEG内の色成分のシーケンスは限定される。

【1051】データトークンの各グループの前には、動きベクトル、量子化器スケール係数等に関する情報を伝達する多くのトークンが置かれる。これらのトークンはピクチャが表示されるようにする必要がないので、無視することができる。

【1052】A. 19. 3. 7 「MPEGピクチャ再整理の効果」

A. 18. 3. 5において説明したように、時間デコーダはMPEGピクチャ再整理を提供する(MPEG reordering=1)ように構成することができる。データストリームにおける次のP/Iピクチャが時間デコーダによりデコーディングがスタートされるまで、P及びIピクチャの出力は遅らされる。時間デコーダの出力時に、新たにデコーディングされたP/Iピクチャのデータトークンは、旧P/Iピクチャからのデータトークンで置き換えられる。

【1053】P/Iピクチャを再整理する時、ピクチャがオフチップピクチャバッファに書き込まれるにつれて、ピクチャのピクチャスタート、時間標準、及びピクチャタイプのトークンが一時的にオンチップに記憶される。ピクチャが表示のために読み出されると、これらの記憶されたトークンが検索される。従って、再整理されたP/Iピクチャはピクチャスタート、時間標準、及びピクチャタイプのための正しいバリューを持っている。

【1054】ピクチャレイヤの下の他の全てのトークンはリオーダーされない。再整理されたP/Iピクチャが表示のために読み出されるにつれて、それはたっいまデコーディングされたばかりのピクチャの低い方のレベルの非データトークンを拾い上げる。このように、これらのサブピクチャレイヤのトークンは無視されるべきである。

【1055】A. 19. 3. 8 「ランダムアクセス及び編集済みシーケンス」

空間デコーダは編集済みMPEGビデオデータ及び、M

P E Gビデオデータへのランダムアクセス後の正しいビデオデコーディングを助ける設備を提供する。

A. 19. 3. 8. 1 「オープンGOP」

ピクチャグループ(GOP)は前のGOPの中のPピクチャから予測されるBピクチャでスタートすることができる。これを「オープンGOP」と呼ぶ。図116はこれを図示している。ピクチャ17及び18は第2のGOPのスタートにあるBピクチャである。GOPが「開かれる」と、エンコーダはPピクチャ16及びIピクチャ19からの予測を用いて、これら2つのピクチャを符号化することができる。あるいはその代わりに、エンコーダはIピクチャ19からのみの予測を用いるように制限することができるであろう。この場合、第2のGOPは「閉鎖GOP」と呼ばれる。

【1056】デコーダが最初のGOPにおいてビデオデコーディングをスタートすると、そのGOPは既にPピクチャ16をデコーディングしているので、そのGOPがたとえ開かれていても、第2のGOPに遭遇しても何の問題も起こらないであろう。しかしながら、デコーダがランダムアクセスを行い、第2のGOPにおいてデコーディングをスタートする場合、それらがP16に依存していれば(つまり、GOPが開かれていれば)、それはB17及びB18をデコーディングできない。

【1057】本発明の空間デコーダがリセットに続く最初のGOPとしてオープンGOPに遭遇するか、あるいはフラッシュトークンを受け取れば、オープンGOPへのランダムアクセスが発生したと仮定される。この場合、ハフマンデコーダは通常の方法でBピクチャのためにデータを消費するであろう。しかしながら、それはIピクチャから離れた(0, 0)動きベクトルで予測されるBピクチャを出力するであろう。その結果、(上記の例における)ピクチャB17及びB18がI19と同一になるであろう。

【1058】この行為はMPEGのV B V規則の正しい維持を確実にする。更に、それはBピクチャが他のデータチャンネルにより予期される出力ストリーム内の位置における出力の中に存在することを確実にする。例えば、MPEGシステムレイヤはオーディオデータに関する表示時間情報をビデオデータに提供する。ビデオ表示時間のスタンプはGOPにおける最初の表示ピクチャ、つまり時間基準0のピクチャを指示する。上記の例では、第2のGOPに対するランダムアクセスの後、最初に表示されるピクチャはB17である。

【1059】ブロークン・クローズドトークンはMPEG closed gopビットを所有する。従って、時間デコーダの出力において、Bピクチャの出力が本物であるか、あるいは「代用品」が空間デコーダによって導入されていたかを判断することが可能である。あるアプリケーションはこれらの「代用品」のピクチャが存在する時、特別な処置を講ずることを希望するかもしれ

れない。

【1060】A. 19. 3. 8. 2 編集済みビデオアプリケーションがMPEGビデオシーケンスを編集する場合、それは2つのGOP間の関係を破壊するかもしれない。編集後のGOPがオープンGOPであれば、それはGOPの始まりにおいてBピクチャを正しくデコーディングすることはもはやできないであろう。MPEGデータを編集するアプリケーションは、編集後のGOPにおいてbroken linkビットを設定し、デコーダにこれらのBピクチャをデコーディングできないことを指示することができる。

【1061】空間デコーダが破壊されたリンクを持つGOPに遭遇した場合、ハフマンデコーダは通常の方法でBピクチャのためにデータをデコーディングするであろう。しかしながら、それはIピクチャから離れた(0、0)動きベクトルで予測されたBピクチャを出力するであろう。その結果、(上記の例における)ピクチャB17及びB18はI19と同一になるであろう。

【1062】ブローケン・クローズドトークンはMPEGのbroken linkビットを所有する。従って、時間デコーダの出力において、Bピクチャの出力が本物であるか、あるいは空間デコーダによって導入されていた「代用品」であるか否かを判断することが可能である。あるアプリケーションはこれらの「代用品」のピクチャが存在する時、特別な処置を講ずることを希望するかもしれない。

セクションA. 20 「後書き込みDRAMインターフェース」

インターフェースは2つの方法で構成できる：インターフェースの詳細なチアミングは種々の異なるDRAMタイプを収容するように構成することができる。

【1063】DRAMインターフェースの「幅」は費用／性能のトレードオフを提供するように構成することができる。

【1064】

【表154】

信号名	入力/出力	説明
DRAM data [31:0]	入/出	32ビット幅のDRAMデータバス。このバスは任意で16または8ビット幅に構成できる。
DRAM addr [10:0]	出	22ビット幅のDRAMインターフェイスアドレスはこの11ビット幅のバスの上に多重送信される時間である。
$\overline{\text{RAS}}$	出	DRAMのローアドレスストロープ信号
$\overline{\text{CAS}} [3:0]$	出	DRAMのカラムアドレスストロープ信号。インターフェイスのデータバスのバイト毎に1信号が提供される。全てのCAS信号が同時に駆動される。
$\overline{\text{WE}}$	出	DRAM書き込み可能信号
$\overline{\text{OE}}$	出	DRAM出力可能信号
DRAM enable	入	この入力信号は、低い場合、インターフェイス上の全ての出力信号を高い電気インピーダンスに持っていく。DRAMインターフェイス上の活動を停止させる。

表A. 20. 1 DRAMインターフェイス信号

【1065】

【表155】

レジスタ名	サイズ/ 方向	リセット 状態	説 明
modify DRAM timing	1ビット rw	0	この機能はレジスタのDRAMインターフェイスタイミング構成レジスタへのアクセスを可能にする。このレジスタがバリュウを0に保持している間、構成レジスタは修正されない。このレジスタに1を書き込むと、構成レジスタを修正するためのアクセスを要求する。このレジスタに0を書き込んだ後、DRAMインターフェイスはタイミングコンフィグレーションレジスタの新しい値を使用する。
page start length	5ビット rw	0	アクセススタートの長さをティックで指定する。使用できる最小バリュウは4である（4ティックを意味する）。0は32ティックの最大長を選択する。
read cycle length	4ビット rw	0	最終ページの読出しサイクルの長さをティックで指定する。使用できる最小バリュウは4である（4ティックを意味する）。0は16ティックの最大長を選択する。

表A. 20. 2 DRAMインターフェイス構成レジスタ (1/5)

[1066]

[表156]

レジスタ名	サイズ/ 方向	リセット 状態	説 明
write cycle length	4ビット rw	0	最終ページの後期書き込みサイクルの長さをティックで指定する。使用できる最小バリュウは4である（4ティックを意味する）。0は16ティックの最大長を選択する。
refresh cycle length	4ビット rw	0	リフレッシュサイクルの長さをティックで指定する。使用できる最小バリュウは4である（4ティックを意味する）。0は16ティックの最大長を選択する。
RAS falling	4ビット rw	0	RASが起こるアクセススタートのスタート後のティック数を指定する。使用できる最小バリュウは4である（4ティックを意味する）。0は16ティックの最大長を選択する。

表A. 20. 2 DRAMインターフェイス構成レジスタ (2/5)

[1067]

[表157]

レジスタ名	サイズ/ 方向	リセット 状態	説 明
CAS falling	4ビット rw	8	読出しサイクル、書き込みサイクル、 あるいは、CASが起るアクセス スタートのスタート後のティック数を 指定する。 使用できる最小バリューは1である (1ティックを意味する)。0は 16ティックの最大長を選択する。
DRAM data width	2ビット rw	0	DRAMインターフェイスデー タバス、D R A M d a t a [31:0]上で使用されるビット 数を指定する。 A. 20. 4を参照。
row address bits	2ビット rw	0	DRAMインターフェイスアドレ スバスのローアドレス部分のために 使用されるビット数を指定する。 A. 20. 5を参照。

表A. 20. 2 DRAMインターフェイス構成レジスタ (3/5)

【1068】

【表158】

レジスタ名	サイズ/ 方向	リセット 状態	説 明
DRAM enable	1ビット rw	1	このレジスタに0のバリューを書き 込むと、DRAMインターフェイス を高い電気インピーダンス状態に押 しやる。DRAMenable信 号が低いか、または0がレジスタに 書き込まれた場合に、0がこのレジ スタから読み込まれる。
refresh inter- val	8ビット rw	0	このバリューは16decoder clockサイクル周期中の リフレッシュサイクル間の間隔を 指定する。1~255までの範囲の バリューが構成できる。バリューが 0であれば、リセット後自動的に ロードされ、有効なリフレッシュ 間隔が構成されるまで、DRAM インターフェイスが連続的に リフレッシュサイクルを実行する ように強制する。refresh intervalは各リセット後 一度だけ構成されるべきである。

表A. 20. 2 DRAMインターフェイス構成レジスタ (4/5)

【1069】

【表159】

レジスタ名	サイズ/ 方向	リセット 状態	説 明
no refresh	1ビット rw.	0	このレジスタにバリュー1を書き 込むと、リフレッシュサイクルの 実行を防止する。
CAS strength	3ビット rw	6	これら3ビットのレジスタは DRAMインターフェイス信号の 出力ドライブ力を形成する。これ は種々の異なるロードのために インターフェイスが形成されるよ うにする。A. 20. 8を参照。
RAS strength			
addr strength			
DRAM data strength			
OEW strength			

表A. 20. 2 DRAMインターフェイス構成レジスタ (5/5)

A. 20. 1 「インターフェースタイミング(tick)」

本発明では、DRAMインターフェースタイミングは装
置の入力クロック率の4倍で動くクロック (decod
er clock) から引き出される。このクロックは
オンチップPLLによって作られる。

【1070】簡潔のために、この高速クロックの周期を
ティックと称する。

【1071】A. 20. 2 「インターフェースオペ
レーション」

インターフェースはDRAMの高速ページモードを使用
する。3つの異なるタイプのアクセスが支持される：

- ・読出し
- ・書込み
- ・リフレッシュ

各々の読出しもしくは書込みアクセスは1つのDRAM
ページアドレスにおいて1～64バイトのバーストを伝
送する。読出し及び書込み伝送は1つのアクセスにおい
て混合されない。各々の連続アクセスは新しいDRAM
ページへのランダムアクセスとして処理される。

【1072】A. 20. 3 「アクセス構造」

各アクセスは2つの部分で構成される：

- ・アクセススタート
- ・データ伝送

各アクセスはアクセススタートによって開始され、その
後1つかそれ以上のデータ伝送サイクルが続く。アクセ
ススタート及びデータ伝送サイクル両方の読出し、書込

み及びリフレッシュの変形がある。

【1073】1アクセスにおける最後のデータ伝送の終
わりに、インターフェースはデフォルト状態になり、
新しいアクセスがスタートする準備ができるまでこの状
態のままである。最後のアクセスが完了した時、新しい
アクセスのスタート準備ができると、その新しいアクセ
スは直ちにスタートする。

【1074】A. 20. 3. 1 「アクセススタート」
アクセススタートは読出しまたは書込み伝送用にページ
アドレスを提供し、幾つかの初期信号状態を設定する。
3つの異なるアクセススタートがある：

- ・リード・スタート
- ・ライト・スタート
- ・リフレッシュ・スタート

各々の場合において、/RAS/のタイミング及びロー
アドレスはレジスタRAS fallingとpage
start lengthによって制御される。/O
E/及びDRAM data [31:0]の状態は、以
前のデータ伝送の終了時から/RAS/になるまで保持
される。3つの異なるタイプのアクセススタートは、/
RAS/になる時に/OE/及びDRAM data
[31:0]を駆動させる方法が異なるだけである。図
118を参照せよ。

【1075】

【表160】

ナン バ ー	特 徴	最 小	最 大	単 位	注
38	レジスタRAS fallingにより 設定されるRAS予備充電周期。	4	16	ティック	
39	レジスタpage start lengthにより設定されるアクセス スタート期間。	4	32		
40	レジスタCAS fallingにより 設定されるCAS予備充電長。	1	16		a
41	レジスタread cycle lengthにより設定される高速 ページ読出しサイクル長。	4	16		
42	レジスタwrite cycle lengthにより設定される高速 ページ書き込みサイクル長。	4	16		
43	WEはCASの1ティック後に起こる。				
44	レジスタrefresh cycleに より設定されるリフレッシュサイクル 長。	4	16		

表A. 20. 3 アクセススタートパラメーター

注a: このバリューはCASがRASリフレッシュが発生する前であることを保証するため、RAS fallingより小さくしなければならない。

A. 20. 3. 2 「データ伝送」

データ伝送サイクルには3つの異なるタイプがある:

- ・高速ページリードサイクル
- ・高速ページレイトライトサイクル
- ・リフレッシュサイクル

リフレッシュスタートの次には1つのリフレッシュサイクルだけが続く。リード（またはライト）スタートの次には、1つかそれ以上の高速ページリード（またはライト）サイクルが続く。

【1076】リードサイクルスタートにおいて、CASがハイとされ、新しいカラムアドレスが駆動される。

【1077】レイトライトサイクルが使用される。/WEは/CAS/の後1ティックだけローとされる。出力データはアドレスの後1ティック駆動される。

【1078】/RAS/の前の/CAS/として、リフレッシュサイクルがリフレッシュサイクルスタートにより始められ、リフレッシュサイクル中は如何なるインタ

ーフェース信号活動もない。リフレッシュサイクルの目的はDRAMが必要とする最低の/RAS/低周期を満たすことである。

30 【1079】A. 20. 3. 3 「インターフェースデフォルト状態」

インターフェース信号はアクセスの終了時にデフォルト状態に入る:

- ・/RAS/、/CAS/、及び/WE/は高い
- ・データ及び/OE/はそれ以前の状態のままである
- ・addrは安定したままである

A. 20. 4 「データバス幅」

2ビットのレジスタDRAM data widthはDRAMインターフェースのデータ通路の幅が構成されるようにする。これは小さなピクチャフォーマットで作業をする時、DRAM費用を最小にする。

【1080】

【表161】

DRAM data width	
0	DRAM data [31:24] 上の 8ビット幅のデータバス
1	DRAM data [31:16] 上の 16ビット幅のデータバス
2	DRAM data [31:0] 上の 32ビット幅のデータバス

表A. 20.4 DRAM data widthの形成

注a. リセット後デフォルト

b. 未使用の信号が高いインピーダンスに保持される。

A. 20.5 「アドレスビット」

オンチップの24ビットのアドレスが作られる。このアドレスがロー及びカラムアドレスを形成するためどのように使用されるかは、データバス幅及びローアドレスのために選択されるビット数により異なる。ある構成では内部のアドレスビットの総ての使用を許さない（従って、「隠れビット」が作られる）。

【1081】ローアドレスはアドレスの中間部分から引き出される。これはDRAMが自然にリフレッシュされる率を最大にする。

【1082】A. 20.5.1 「低オーダーのカラムアドレスビット」

64バイトまでの高速ページモード伝送用のアドレスを提供するために、最も重要でない4～6ビットのカラムアドレスが使用される。これらの伝送を制御するために必要なアドレスビット数はデータバスの幅により異なる。（A. 20.4を参照）

A. 20.5.2 「ローアドレスビット」

ローアドレスを提供するため24ビットの内部アドレスの中間セクションから取られるビット数は、レジスタ row address bitsによって形成される。

【1083】

【表162】

row address bits	ローアドレスの幅
0	9ビット
1	10ビット
2	11ビット

表A. 20.5 row address bitsの形成

使用されるローアドレスの幅は使用されるDRAMのタイプ、及びローアドレスのMSBがDRAMの多重バンクにアクセスするためのデコーディングされたオフチップであるか否かにより異なる。

【1084】注：ローアドレスは内部アドレスの中間から引き出される。ローアドレスのあるビットがDRAM

バンクを選択するためにデコーディングされる場合、これらの「バンクセレクトビット」の全ての可能なバリエーションがDRAMバンクを選択しなければならない。そうでなければ、アドレス空間に空孔が残されることになる。

【1085】

【表163】

row address bits	ローアドレスビット	バンクセレクト	DRAMの深さ
0	DRAM addr [8:0]		256K
1	DRAM addr [8:0]	DRAM addr [9]	256K
	DRAM addr [9:0]		512K
	DRAM addr [9:0]		1024K
2	DRAM addr [8:0]	DRAM addr [10:9]	256K
	DRAM addr [9:0]	DRAM addr [10]	512K
	DRAM addr [9:0]	DRAM addr [10]	1024K
	DRAM addr [10:0]		2048K
	DRAM addr [10:0]		4096K

表A. 20. 6 row address bits用のバリュースelect

A. 20. 6 「DRAMインターフェース可能化(enable)」

DRAMインターフェース上の全ての出力信号を高インピーダンスにするには2通りの方法がある。DRAM enableレジスタ及びDRAM enable信号である。レジスタ及び信号共DRAMインターフェースが操作するためロジック1でなければならない。いずれかが低ければ、インターフェースが高インピーダンスにされ、インターフェースを介してのデータ伝送が停止される。

【1086】DRAMインターフェースを高インピーダンスにもっていく能力は、空間デコーダ（もしくは時間デコーダ）が使用されていない時に、他の装置が空間デコーダ（もしくは時間デコーダ）によって制御されるDRAMをテストまたは使用できるようにするために提供され、他の装置が通常のオペレーション中にメモリーを共有できるようにするためではない。

【1087】A. 20. 7 「リフレッシュ」

レジスタno refreshに書き込むことにより不能化されない限り、DRAMインターフェースはレジスタrefresh intervalにより決定される間隔で、/RAS/リフレッシュサイクルの前の/CAS/を使用して、DRAMを自動的にリフレッシュする。

【1088】refresh intervalのバリュースelectは16decoder clockサイクルの期間におけるリフレッシュサイクル間の間隔を指定する。1～255の範囲のバリュースelectを選択することができる。バリュースelect 0はリセット後自動的にロードされ、（一度イネーブルとされると）有効なリフレッシュ間隔が形成されるまで、DRAMインターフェースが連続してリフレッシュサイクルを実行するように強制する。refresh intervalが各リセット後一度だけ形成されるようにすることが薦められる。

【1089】A. 20. 8 「信号強度」

DRAMインターフェースの出力の駆動力は3ビットレジスタ/CAS/strength、/RAS/strength、addr strengthを用いて、ユーザーにより形成される。この3ビットバリュースelectのMSBは高速もしくは低速のエッジ率のいずれかを選択する。2つの重要性の低いビットが異なるロードキャパシタンスのための出力を形成する。

【1090】リセット後のデフォルト強度は6であり、12pFでロードされる場合、GNDとVDD間の駆動信号に対しておよそ10nsを取る出力を形成する。

【1091】

【表164】

強度バリュー	駆動特性
0	6 p f ロードにおよそ4 ns/V
1	12 p f ロードにおよそ4 ns/V
2	24 p f ロードにおよそ4 ns/V
3	48 p f ロードにおよそ4 ns/V
4	6 p f ロードにおよそ2 ns/V
5	12 p f ロードにおよそ2 ns/V
6*	24 p f ロードにおよそ2 ns/V
7	48 p f ロードにおよそ2 ns/V

表A. 20.7 出力強度形成

注a. リセット後デフォルト

およそ駆動中のロードのために出力が形成される時、それは表168～表169に明記されるAC電気特性を満たすであろう。適切に形成された場合、各出力はその負荷にほぼ適合し、従って最小のオーバーシュートが信号遷移後に発生するであろう。

【1092】A. 20.9 「リセット後」

リセット後、DRAMインターフェース形成レジスタは全てそれらのデフォルトバリューにリセットされる。これらのデフォルト構成の内最も重要なものを以下に記す：

・DRAMインターフェースが不能化(disable)され、高インピーダンスに行くことが許される。

【1093】・リフレッシュ間隔が特別なバリュー0に形成され、それはインターフェースが再可能化された後、連続的なリフレッシュサイクルの実行を意味する。

【1094】・DRAMインターフェースはその最低速の形成に設定される。

【1095】ほとんどのDRAMは電力が最初に印加され、その後通常オペレーションが可能となる前に多数のリフレッシュサイクルが続いた後、100 μ s ～ 500 μ s 間の「休止」を要求する。

【1096】リセット後直ちに、DRAMインターフェースはDRAM enable信号及びDRAM enableレジスタの両方が設定されるまで不活性化される。これらが設定された時、DRAMインターフェースはDRAMインターフェースが形成されるまで、(使用されるクロック周波数に応じて、ほぼ400 ns 毎に)リフレッシュサイクルを実行するであろう。

【1097】ユーザーには、power up後の、また必要な数のリフレッシュサイクルがデータ伝送を試みる前に発生したことを確かめるため、DRAMインターフェースを可能化した後充分時間があるように、DRAMの「休止」を確実にするための責任がある。

【1098】リセットを宣言する間は、DRAMインターフェースはDRAMをリフレッシュすることができない。しかしながら、デコーダチップが要求するリセットタイムは、それらをリセットでき、その後DRAMの内容が崩壊する前にDRAMインターフェースを再可能化することができる程度に充分短くなっている。これはデバッギング中に必要とされるかもしれない。

【1099】

【表165】

記号	パラメーター	最低	最大	単位
V_{DD}	GNDに対する供給電圧	-0.5	6.5	V
V_{IN}	ピン上の入力電圧	GND+0.5	$V_{DD}+0.5$	V
T_A	オペレーション温度	-40	+85	℃
T_S	保管温度	-55	+150	℃

表A. 20.8 最大定格

記号	パラメーター	最低	最大	単位
V_{DD}	GNDに対する供給電圧	4.75	5.25	V
GND	接地	0	0	V
V_{IH}	入力ロジック「1」電圧	2.0	$V_{DD} + 0.5$	V
V_{IL}	入力ロジック「0」電圧	$GND + 0.5$	0.8	V
T_A	オペレーション温度	0	70	°C

表A. 20. 9 DCオペレーション状態

注 a. TBA直線 t/min 横方向の空気流を伴う。

【1101】

【表167】

記号	パラメーター	最低	最大	単位
V_{OL}	出力ロジック「0」電圧		0.4	V
V_{OH}	出力ロジック「1」電圧	2.8		V
I_O	出力電流	± 100		μA
I_{OZ}	出力オフ状態漏出電流	± 20		μA
I_{LZ}	入力漏出電流	± 10		μA
I_{CO}	RMS電源電流		500	mA
C_{IN}	入力キャパシタンス		5	pF
C_{OUT}	出力/IOキャパシタンス		5	pF

表A. 20. 10 DC電気特性

- 注 a. ACパラメーターは測定レベルとして $V_{OLmax} = 0.8V$ を用いて指定される。
- b. これはインターフェイスの定常状態の駆動能力である。過渡的電流はそれより大きいかもしれない。

A. 20. 10. 1 「AC特性」

【表168】

【1102】

ナンバー	パラメーター	最低	最大	単位	注
45	サイクルタイム、例えば t_{PC}	-2	-2	ns	
46	サイクルタイム、例えば t_{RC}	-2	-2	ns	
47	t_{PR} , t_{CP} , t_{CPN} 等の高パルス	-5	+2	ns	
48	t_{RAS} , t_{CAS} , t_{CAC} , t_{WP} , t_{RASP} , t_{RASC} 等の低パルス	-11	+2	ns	
49	t_{ACP}/t_{CPA} 等のサイクルタイム	-8	+2	ns	

表A. 20. 11 ストローブ用の公称バリュウからの差

注 a. 信号のドライバー力はその負荷のために適切に形成されなければならない。

[1103]

[表169]

ナンバー	パラメーター	最低	最大	単位	注*
50	tRCD, tCSR等のストロープからストロープへの遅れ	-3	+3	ns	
51	tRSH, tCHS, tRWL, tCWL, tRAC, tOAC/OE, tCHR等の低いホールドタイム	-13	+3	ns	
52	tCRP, tRCS, tRCH, tRRH, tRPC等のストロープからストロープへの予備充電	-8	+3	ns	
	tCP等の広いDRAM上の2つのCAS信号間、もしくはtRPC等のRAS上昇とCAS降下間のCAS予備充電パルス	-5	+2	ns	
	不能化前の予備充電、例えばtRHCP/CPRH	-12	+3	ns	

表A. 20. 12 2つのストロープ間の公称バリュウからの差

注a. 2つの信号のドライバ電力はその負荷のために適切に形成されなければならない。

セクションB. 1 「スタートコード検出器」

B. 1. 1 「展望」

図18において前述したように、スタートコード検出器(SCD)は空間デコーダ上の最初のブロックである。その基本的な目的は入力データストリームの中でMPEG、JPEG及びH. 261のスタートコードを検出し、それらを関連トークンと置き換えることである。それは更にマイクロプロセッサインターフェースを介して入力データストリームへのユーザーアクセスを可能にし、トークンデータストリームの予備フォーマッティング及び「整頓」を遂行する。SCDは生のバイトデータもしくはトークンフォーマットに既に組み立てられたデータのいずれかを受け取ることができることを思いだし

てほしい。

【1104】典型的に、スタートコードはMPEG、H. 261、及びJPEGに対して、各々24、16及び8ビットである。スタートコード検出器はマイクロプロセッサインターフェース(upi)もしくはトークン/バイトポートのいずれかからバイトで入力データを受け取り、それを3つのシフトレジスタを通して移動させる。第1のレジスタは8ビットの並直列アウトであり、第2のレジスタはプログラム可能な長さ(16もしくは24ビット)で、スタートコードが検出される場所であり、第3のレジスタは15ビット幅で、データを15ビットのトークンに再フォーマッティングするために使用される。更に、第2及び第3のSRと並列に動く2つの「タグ」シフトレジスタ(SR)がある。これらはデータSR内で連合するビットが良いか悪いかを指示するた

めのタグを含む。データトークンの一部ではなくSCDにより認識されない入力バイトが、シフトレジスタをバイパスできるようにし、3つの全てのシフトレジスタがフラッシュされ(空になり)、内容が首尾良く出力された時に、出力される。認識された非データトークンはSCD、スプリングトラップ、セットフラグを形成するために使用される。更にそれらはシフトレジスタをバイパスし、変更されずに出力される。

【1105】B. 1. 2 「主ブロック」

スタートコード検出器のためのハードウェアは10個のステートマシンで構成される。

【1106】B. 1. 2. 1 「入力回路(scdipc.sch. implm. M)」

入力回路は3つのオペレーションモード: トークン、バイト及びマイクロプロセッサインターフェースを持つ。これらのモードはデータが生のバイトストリーム(であるが2線式インターフェースを使用する)ものとして、トークンストリームとして、あるいはupiを介したユーザーによって入力されるようにする。全ての場合に、入力回路は適切な場合にデータトークンヘッダを発生させることにより、常に正しいデータトークンを出力する。upiモードへのからの遷移はシステムクロックに同期化され、upiはアクセスを得る前のデータストリームにおける安全ポイントまで待たされる。バイトモードピンは入力回路がトークンモードであるか、もしくはバイトモードであるかを決定する。更に、どの基準が解読されているかに関するシステムに対する初期通知(従ってコーディングスタンダードトークンが発せられ

る) が3モードのいずれかにおいて行われ得る。

B. 1. 2. 2 「トークンデコーダ (s c d i p n e
w. s c h, s c d i p n e m. M)」

このブロックは入力トークンをデコーディングし、他の

ブロックにコマンドを発する。

{1107}

[表170]

入力トークン	出力コマンド	コメント
NULL	待機せよ	ゼロが取り除かれる。
DATA	通常	次のバイトを第一のSRにロードする。
CODINGSTD	バイパスせよ	シフトレジスタをフラッシュし、パディングを実行し、出力し、バイパスモードに切り換える。CODINGSTANDARDレジスタをロードする。
FLUSH	バイパスせよ	SRをパディングによりフラッシュし、出力し、バイパスモードに切り換える。
ELSE (認識されない トークン)	バイパスせよ	SRをパディングによりフラッシュし、出力し、バイパスモードに切り換える。

表B. 1. 1 認識された入力トークン

注：コーディングスタンダードにおける変化はSRがフラッシュされた後、二線インターフェースを介して全てのブロックに送られる。これは1つのデータストリームから別のデータストリームまでの変化がSCD中ずっと正しいポイントで発生することを確実にする。この原則はコーディングスタンダードにおける変化が新しいストリームの前に全チップを通して流れることができるように、表示中ずっと適用される。

[1108] B. 1. 2. 3 「JPEG (s c d j p
e g. s c h, s c d j p e g m. M)」

switch (state)

{

case (LOOKING) :

if (input=0xff)

{

state=GETVALUE; /*Found a marker*/

remove; /*Marker gets removed*/

}

state=LOOKING;

break;

case (GETVALUE) :

if (input=0xff)

{

state=GETVALUE; /*Overlapping
markers*/

remove;

}

else if (input=0x0-0)

JPEGにおけるスタートコード (マーカー) は充分異なっており、JPEGは全て自身に対するステートマシンの持っている。本発明において、このブロックがJPEGマーカー検出、長さのカウント/チェック、及びデータの除去の全てを処理する。検出されたJPEGマーカーはスタートコード (v not tを持つ、後述のテキスト参照) としてフラグ表示され、s c d i p n e wからのコマンドはオーバーライドされ、バイパスするように強いられる。オペレーションはコードで最も良く説明される。

```

    state=LOOKING; /*Wasn't a marker/
    insert (0xff); /*Put the 0xff
                        back/
}
else
{
    command=BYPASS; /*override command/
    if (lc) /*Dose the marker have a
        length count/
        state=GETLC0;
    else
        state=LOOKING;
    break;
case (GETLC0) :
    loadlc0; /*Load the top length count
        byte/
    state=GETLC1;
    remove;
    break;
case (GETLC1)
    loadlc1;
    remove;
    state=DECLC;
    break;
case (DECLC) :
    lcnt=lcnt-2
    state=CHECKLC;
    break;
case (CHECKLC) :
    if (lcnt=0)
        state=LOOKING; /*No more to do/
    else if (lcnt<0)
        state=LOOKING; /*generate Illegal
                        Length Error/
    else
        state=COUNT;
    break;
case (COUNT) :
    decrement length count until 1
    if (lc<=1)
        state=LOOKING;
}

```

B. 1. 2. 4 「入力シフター (scinshft. sch, scinsh. M)」

このブロックの基本オペレーションは全く簡単である。このブロックは入力回路からデータバイトを取り出し、シフトレジスタにロードし、それをシフトする。しかしながら、それは入力デコーダからのコマンドにも従い、バイパスモードへの／からの遷移を処理する（他のSR 50

をフラッシュする）BYPASSコマンドを受け取ると、連合するバイトはシフトレジスタにロードされない。その代わりに、「ラビッシュ(rubbish)」(タグ=1)がシフトされ、他のシフトレジスタに保持されているデータを出力させるように強いる。その後、ブロックはこの「ラビッシュ」がトークン発生器に現れたことを示す「フラッシュされた」信号を待つ。入力バイトは次

にトークン発生器に直接送られる。

【1109】B. 1. 2. 5 「スタートコード検出器 (scdetect. sch, scdetm. M)」

このブロックは16もしくは24ビットのスタートコード検出ロジック及び「有効な内容」検出ロジックにプログラム可能な2つのシフトレジスタを含む。MPEGスタートコードは完全な24ビットを要求するが、H. 261は16ビットだけを要求する。

【1110】本発明では、第1のSRがデータ用であり、SRには(二線インターフェースの意味で)ギャップもしくは失速はないが、第2のSRはデータSR内のビットが有効か否かを指示するタグを有するが、それらが包含するビットはフラッシュされる一方、無効(ラビッシュ)であってもよい。スタートコードが検出されると、タグシフトレジスタのビットが検出器SRの内容を無効にするために設定される。

【1111】スタートコードはSRの内容が全て有効にならない限り検出され得ない。非バイト整列スタートコードが検出され、フラグ表示されてもよい。更に、スタートコードが検出されると、重なっているスタートコードが調べ上げられるまで非整列スタートコードは明確にフラグ表示され得ない。この機能を果たすため、検出されたスタートコード(バイトがそれに続く)の「バリュ

ー」がscinshiftとscdetectを通して

右に、そしてscoshiftシフトされる。別のスタートコードを検出せずにscoshiftに達すると、除去されたのは重なっているスタートコードであり、それは有効なスタートコードとしてフラグ表示される。

【1112】B. 1. 2. 6 「出力シフター (scoshift. sch, scoshm. M)」

出力シフターの基本オペレーションはscdetectから直列データ(及びタグ)を取り出し、それを15ビットのワードに詰め、出力することである。他の機能は以下の通りである：

B. 1. 2. 6. 1 「データバディング」
出力は15ビットワードで構成されるが、入力は任意の数のビットで構成できる。従って、フラッシュするためには、最後のワードを15ビットまでにするためにビットを追加する必要がある。これら追加ビットはバディングと呼ばれ、ハフマンブロックによって認識され除去されねばならない。バディングは次のように定義される：

最後のデータビットの後、「ゼロ」が挿入され、15ビットワードを作り上げるために充分な「もの」がそれに続く。

【1113】バディングを含むデータワードはデータ

トークンの終わりであることを指示するために、低拡張ビットで出力される。

【1114】B. 1. 2. 6. 2 「フラッシュの発生」

本発明によれば、「フラッシュ」オペレーションの発生

は、全てのSRがフラッシュされる時を検出し、それを入力シフターに合図することを含む。入力シフターにより挿入された「ラビッシュ」が出力シフターのエンドに達し、出力シフターがそのバディングを完了すると、

「フラッシュされた」信号が発せられる。この「フラッシュされた」信号は入力シフターがバイパスモードに入ることが安全である前に、トークン発生器を通過しなければならない。

【1115】B. 1. 2. 6. 3 「有効なスタートコードのフラグ表示」

scdetectがスタートコードを発見したことを指示すると、バディングが遂行され、現在のデータが出力される。スタートコードバリュー(次のバイト)が検出器を通してシフトされ、重なっているスタートコードを除去する。別のスタートコードが検出されずに「バリュー」が出力シフターに達すると、それは重ねられず、そのバリューはそれがスタートコードバリューであることを示すフラグv not t (ValueNotToken)を付けて送られる。しかしながら、別のスタートコードが(scdetectにより)検出される一方、出力シフターがそのバリューを待っていると、オーバーラッピングエラーが発せられる。この場合、第1のバリューは捨てられ、システムは第2のバリューを待つ。このバリューも重ねることができ、こうして非重複スタートコードが見つかるまで同じ手順が繰り返されるようにする。

【1116】B. 1. 2. 6. 4 「スタートコードの後の整頓(tyding up)」

適切なスタートコードを検出し出力した後、(ラビッシュでない)データが到着し始めると、新しいデータヘッダが作られる。

【1117】B. 1. 2. 7 「データストリーム発生器 (sctokrec. sch, sctokrem. M)」

データストリーム発生器は、1つはバイパスされたトークンのためのscinshiftから、他の1つは詰め込まれたデータ及びスタートコードのためのscoshiftからの二線インターフェース入力を持つ。2つのソース間の切り替えは、(いずれかのソースからの)現在のトークンが(到着した低拡張ビットを)完了した後にのみ許される。

【1118】B. 1. 2. 8 「数変換をスタートするためのスタートバリュー (scdromhw. sch, schrom. M)」

スタートバリューをトークンに変換するプロセスは2段階で行われる。このブロックは520の奇数の位置コードを16のコーディングスタンダードの独立インデックスに減少させるコーディングスタンダード依存問題点を主に扱う。初期に述べたように、(JPEGのバリューを含む)スタートバリューは他の全てのデータから

フラグ (value not token) によって識別される。value not token が高ければ、このブロックは4または8ビットバリューをコーディングスタンダードに応じて、スタンダードから独立している4ビットのs

start number に変換し、未認識のスタートコードをフラグ表示する。スタート数は次の通りである：

【1119】

【表171】

スタート/ マーカークード	インデックス (startnumber)	結果として生じる トークン
not a start code	0	
sequence start code	1	SEQUENCE START
group start code	2	GROUP START
picture start code	3	PICTURE START
slice start code	4	SLICE START
user data start code	5	USER DATA
extension start code	6	EXTEN- SION DATA

表B. 1. 2 スタートコード数 (インデックス) (1/3)

【1120】

【表172】

スタート/ マーカークード	インデックス (startnumber)	結果として生じる トークン
sequence end code	7	SEQUENCE END
JPEGマーカークード		
DHT	8	DHT
DQT	9	DQT
DNL	10	DNL
DRI	11	DRI
MPEG/H. 261用のトークンにマッピングできるJPEGマーカークード		
SOS	picture start code	PICTURE START
SOI	sequence start code	SEQUENCE START
EOI	sequence end code	SEQUENCE END

表B. 1. 2 スタートコード数 (インデックス) (2/3)

【1121】

【表173】

スタート/ マーカークード	インデックス (start number)	結果として生じる トークン
SOF0	group start code	GROUP START
拡張もしくはユーザーデータを作り出すJ PEGマーカークード		
JPG	extension start code	EXTENSION DATA
JPGn	extension start code	EXTENSION DATA
APPn	user data start code	USER DATA
COM	user data start code	USER DATA
注: 認識されない全てのJ PEGマーカークードはextn start codeインデックスを作り出す。		

表B. 1. 2 スタートコード数 (インデックス) (3/3)

B. 1. 2. 9 「トークン変換に対するスタート数
(sconvert. sch、sconverm. M)」

変換の第2段階において上記スタート数 (もしくはイン
デックス) がトークンに変換される。このブロックは更
に拡張及びユーザーデータを適切に捨てるトークン拡張
及びサーチモードを処理する。

【1122】サーチモードはランダムポイントでデータ
ストリームに入る手段である。サーチモードは8つのバ
リュウの1つに設定できる:

0 : 通常のオペレーション - 次のスタートコード
を見つける。

【1123】1/2: システムレベルサーチは空間デ
コードに関しては実施されない。

【1124】3 : シーケンス以上のものを探す

4 : グループ以上のものを探す

5 : ピクチャ以上のものを探す

6 : スライス以上のものを探す

7 : 次のスタートコードを探す

非ゼロサーチモードは所望のスタートコード (またはシ

```
switch (input data)
```

```
case (FLUSH)
```

```
1. if (in picture)
```

```
output=PICTURE END
```

```
2. output=FLUSH
```

```
3. if (in picture
```

```
& stop after picture)
```

```
sap error=HIGH
```

```
in picture=FALSE;
```

```
4. in picture=FALSE;
```

```
break
```

```
case (SEQUENCE START)
```

ンタックスにおいてより高い) が検出されるまで、デー
タが捨てられるようにする。

【1125】このブロックは更にPICTURE及びS
LICEスタートトークンにトークン拡張を加える。

【1126】・ピクチャスタートはPICTURE N
UMBER、ピクチャの4ビットカウントで拡張され
る。

【1127】・スライススタートはs v p (スライス垂
直位置) で拡張される。これはスタートコードマイナス
1 (MPEG、H. 261)、及びマイナスOXDO
(J PEG) の「バリュウ」である。

30 【1128】B. 1. 2. 10 「データストリームフ
ォーマット (scinsert. sch、sci
nserx. M)」

本発明では、データストリームフォーマットはピ
クチャエンド、フラッシュ、コーディングスタンダー
ド、シーケンススタートのトークンの条件付き挿入、及
びSTOP AFTER PICTUREイベントの発
生に関する。その機能はソフトウェアにおいて最も良く
簡略化され説明できる:

363

364

```

1. if (in picture)
  output=PICTURE END
2. if (in picture
    & stop after picture)
  2a. output=FLUSH
  2b. sap error=HIGH
    in picture=FALSE
3. output=CODING STANDARD
4. output=standard
5. output=SEQUENCE START
6. in-picture=FALSE;
break
case (SEQUENCE END)
case (GROUP START) :
  1. if (in picture)
    output=PICTURE END
  2. if (in picture
    & stop after picture)
    2a. output=FLUSH
    2b. sap error=HIGH
      in picture=FALSE
  3. output=SEQUENCE END or
    GROUP START
  4. in-picture=FALSE;
break
case (PICTURE END)
  1. output=PICTURE END
  2. if (stop after picture)
    2a. output=FLUSH
    2b. sap error=HIGH
  3. in-picture=FALSE
break
case (PICTURE START)
  1. if (in picture)
    output=PICTURE END
  2. if (in picture
    & stop after picture)
    2a. output=FLUSH
    2b. sap error=HIGH
  3. if (insert sequence start)
    3a. output=CODING START
    3b. output=standard
    3c. output=SEQUENCE START
      insert sequence start=FALSE
  4. output=PICTURE START
    in picture=TRUE
break
default: Just pass it through

```

セクションB. 2 「ハフマンデコーダ及びパーザー(p B. 2. 1 「概論」

arser)」

50 本セクションは本発明によるハフマンデコーダ及びパー

ザ回路について説明する。

【1129】図127はハフマンデコード及びパーザの高レベルのブロック線図を示す。明瞭さのために、本線図では多くの信号及びバスを省略しており、特にデータが(図示された大きなループ内で)後方に送られる場所が幾つかある。

【1130】本質的に、本発明のハフマンデコード及びパーザは、プログラム可能ステートマシンによって制御される(線図の底部に沿って示される)多くの専用処理ブロックで構成される。

【1131】データは「インシフト」ブロックにより符号化データバッファから受け取られる。この時点で、本質的に遭遇するであろう2つのタイプの情報がある: データトークンが所有する符号化データ、及びスタートコード検出器により各々のトークンと既に置き換えられているスタードコードである。他のトークンと遭遇する可能性もあるが、(データトークン以外の)全てのトークンは同じ方法で処理される。トークン(スタートコード)は(H. 261、JPEGもしくはMPEGにおいて)多数のデータがこれから符号化されるにつれて、特別な場合として処理される。

【1132】本発明では、データトークンが所有する全てのデータが直列形態(ビットバイビット)でハフマンデコードに伝送される。もちろん、このデータはハフマン符号化されない多くのフィールドを含むが、固定長により符号化される。それにもかかわらず、このデータはハフマンデコードに直列に送られる。ハフマン符号化データの場合、ハフマンデコードは実際のハフマンコードがインデックス数で置き換えられるデコーディングの第1ステージだけを実行する。解説される特別なコード表にNディストリクトハフマンコードがある場合、この「ハフマンインデックス」の範囲は0~N-1である。更に、ハフマンデコードは"no op"、つまり「ノーオペレーション」モードを有し、それはハフマンデコードによる処理なしに、データまたはトークン情報に沿って次のステージに送られるようにする。

【1133】データユニットインデックス部はテーブルルックアップオペレーションを実行する比較的簡単な回路ブロックである。それはハフマンデコーディングプロセスの第2ステージからその名前を取り出し、そこでハフマンデコードにおいて得たインデックス数が簡単なテーブルルックアップによって実際の解説データに変換される。データユニットインデックス部はハフマンデコードと協働して1つの論理ユニットとして作用する。

【1134】ALUが次のブロックであり、解説済みデータに他の変換を加えるために提供される。データユニットインデックス部は比較的任意のマッピングに適しており、ALUは算術がよりふさわしい場所に使用できる。ALUはデコーディングアルゴリズムの様々な部分を実施するために操作できるレジスタファイルを具備す

る。特に、ベクトル予測及びDC予測を保持するレジスタがこのブロックに含まれる。ALUはオペランドセレクションロジックを持つ単純なアダーのまわりに基礎づけられる。更に、それはサイン拡張タイプのオペレーション専用回路を含む。シフトオペレーションが実行されやすいが、これはシリアル様式で遂行され、パレルシフトは存在しないであろう。

【1135】本発明によるトークンフォーマット部は、ビデオパーザ内の最後のブロックであり、解説済みデータを残りのデコードに送ることができるトークンに最終的に組み込む仕事をする。この時点で、この特定のピクチャ用のデコードにより使用されるであろう多くのトークンがある。

【1136】8ビット幅であり、2線式インターフェースと共に使用されているパーザステートマシンは、他のブロックのオペレーションを調整する仕事をする。本質的に、それは非常に単純なステートマシンであり、他のブロックに送られる非常に広範囲の「マクロコード」制御ワードを作り出す。図127は指令ワードがブロックからブロックへとデータの側を通過する。これが実情であり、異なるブロック間の伝送が2線式インターフェースによって制御されることを理解することが重要である。

【1137】本発明では、ビデオパーザ内の各ブロック間に2線式インターフェースがある。更に、ハフマンデコードはシリアル、データと共に、また制御トークンと共に作動し、インシフトは一度に1ビットずつデータを入力する。従って、2つのモードのオペレーションがある。データがデータトークンを介してハフマンデコードに入力されると、そのデータは一度に1ビットずつシフトを通過する。更に、インシフトとハフマンデコードの間に2線式インターフェースがある。しかしながら、他のトークンは一度に(シリアルに)1ビットの形態でシフトされず、むしろトークンのヘッダにおいてシフトされる。データトークンが入力されると、アドレス情報を含むヘッダが削除され、そのアドレスの次のデータが一度に1ビットでシフトされる。データトークンではない場合、全トークン、ヘッダ及び全てのものが一度に全てハフマンデコードに表示される。

【1138】本発明では、ビデオパーザ用の2線式インターフェースは2つの有効なラインを持つ点で異例のことであることを理解することが重要である。1つのラインはシリアル式に有効で、他方のラインはトークン式に有効である。更に、両ライントークンが同時に断定することはできないかもしれない。片方のラインが断定されるか、あるいは有効なデータが存在しなければ、2つの有効ラインがあってもどちらも断定できないこともあり、他の方向には1つだけのアクセプトワイヤがあることを認識すべきである。しかしながら、これは問題とはならない。ハフマンデコードは現在のシンタックスに基

づいて次に為されるべき必要に応じて、シリアルデータもしくはトークン情報のいずれを望んでいるかを知ることができる。従って、有効なアクセプト信号がそれに依りて設定され、Acceptがハフマンデコードからインシフトに送られる。適切なデータもしくはトークンが存在すれば、インシフトは有効な信号を送る。例えば、典型的な指令はハフマンコードを解説し、それをデータユニットインデックス部において変換し、その結果をALUにおいて修正することができ、この結果がトークンワードに形成される。これを実施するための全ての情報を含む1つのマイクロコード指令ワードが作られる。コマンドはハフマンデコードに直接送られ、ハフマンデコードは完全な記号を解説し終えるまで、「インシフト」ブロックからデータビットを1つずつ要求する。制御トークンは並列に入力される。これが一度発生すると、解説されたインデックスバリューは元のマイクロコードワードと共にデータユニットインデックス部に送られる。ハフマンデコードはこのオペレーションを実行するために数個のサイクルを必要とし、実際、サイクル数は解説されるデータによって決定される。データユニットインデックス部は次にマイクロコード指令ワードにおいて特定された表を用いて、このバリューをマッピングする。このバリューは元のマイクロコードワードと共に再び次のブロック、ALUに送られる。ALUが適切なオペレーションを完了する（サイクル数はやはりデータ従属であるかもしれない）と、ALUはトークンワードを形成する方法を制御するマイクロコードワードと共に、トークンフォーマットブロックに適切なデータを送る。

【1139】ALUは多数のステータスワイヤもしくはパーザーステートマシンに送り返される「条件コード」を持つ。これはステートマシンが条件付きジャンプ指令を実行できるようにする。事実、全ての指令は条件付きジャンプ指令である：選択できる条件の1つがバリュー“False”に配線される。この条件を選択することにより、「ノージャンプ」指令を構成することができる。

【1140】本発明によれば、トークンフォーマット部は2つの入力：ALUからのデータフィールド及び／もしくはパーザーステートマシンから入る一定フィールドを持つ。それに加えて、1つのソースからどの程度のビットを取り、全体で8ビットのために他のソ-

スからの残りのビットで満たすべきかをトークンフォーマット部に告げる指令がある。例えば、HORIZONTAL SIZEはHORIZONTAL SIZEトークンとしてそれを特定する不変のアドレスである8ビットフィールドを持つ。この場合、8ビットは一定のフィールドから得られ、ALUから得られるデータはない。しかしながら、それがデータトークンであれば、一定のフィールドから6ビットを容易に得られるであろうし、低い方の2ビットはALUからの色成分を示す。従って、トークンフォーマット部はこの情報を得、それを残りのシステムで使用するためにトークンの中に置く。上記の例において、各ソースからのビット数は単に説明目的のためであり、当業者であればいずれのソースからのビット数をも変更できることを認識するであろうということに注目すべきである。

【1141】ALUはピクチャの構造を通してカウントするために使用されるカウンタバンクを含む。ピクチャのディメンションは、レジスタバンクの一部として「マイクロプログラマ」に現れるカウンタと連合するレジスタにプログラムされる。幾つかの条件コードはこのカウンタバンクから出力され、それは「ピクチャ・スタート」、「マクロブロック・スタート」等に基づく条件付きジャンプを可能にする。

【1142】注意すべきことは、パーザーステートマシンも「デマルチブレックスステートマシン」と称されることである。両用語が本文書において使用される。

【1143】入力シフト

本発明では、入力シフトは2つのパイプラインステージデータバス“hfidp”と制御Zcells“hfi”で構成される非常に単純な回路である。

【1144】第1のパイプラインステージでは、トークンデコーディングが起こる。このステージにおいて、データトークンだけが認識される。データトークンに含まれるデータは一度に1ビットだけハフマンデコードにシフトされる。第2のパイプラインステージはシフトレジスタである。データトークンのラストワードにおいて、符号化データバッファを通して任意のビット数を伝達することができるように、特別なコーディングが発生する。ラストデータワードにおいて可能な全てのパターンを以下に記す。

【1145】

【表174】

E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	ビット数
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	ゼロ
x	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
x	x	0	1	1	1	1	1	1	1	1	1	1	1	1	2
x	x	x	0	1	1	1	1	1	1	1	1	1	1	1	3
x	x	x	x	0	1	1	1	1	1	1	1	1	1	1	4
x	x	x	x	x	0	1	1	1	1	1	1	1	1	1	5
x	x	x	x	x	x	0	1	1	1	1	1	1	1	1	6
x	x	x	x	x	x	x	0	1	1	1	1	1	1	1	7
x	x	x	x	x	x	x	x	0	1	1	1	1	1	1	8
x	x	x	x	x	x	x	x	x	0	1	1	1	1	1	9
x	x	x	x	x	x	x	x	x	x	0	1	1	1	1	10
x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	11
x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	12
x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	13
x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	14

表B. 2. 1 ラストデータワードにおいて可能なパターン

シフトレジスタにおいてデータビットが左に1つずつシフトされるにつれて、ビットパターン「0に続く全ての数」が探される（パディング）。これはシフトレジスタにおける残りのビットが有効ではなく、捨てられることを示す。この動作はデータトークンのラストワードにおいてのみ発生することに注目。

【1146】前述したように、他の全てのトークンはハフマンデコードに並列で送られる。それらは第2のパイプラインステージにもロードされるが、シフトは発生しない。データヘッダは捨てられ、ハフマンには全く送られないことに注目。2つの「有効な」ワイヤ（out valid及びserial valid）が提供される。1つだけが所定の時間に断定され、それはどのタイプのデータがその瞬間に表示されるかを指示する。

【1147】B. 2. 2 「ハフマンデコード」

ハフマンデコードは多数のオペレーションモードを持つ。最も顕著なものはそれがハフマンコードを解読し、それらをハフマンインデックス数に換えることである。それに加えて、ハフマンデコードは指令ワードにより決定される長さの固定長コードを（ビットに）解読することができる。ハフマンデコードは更にインシフトブロックからのトークンを受け取ることができる。

【1148】ハフマンデコードは非常に小さなステートマシンである。これはブロックレベルの情報を解読する時に使用される。これはパーザーステートマシンが決定をするのに長くかかりすぎるからである（なぜな

ら、それはデータに関する決定をし、新しいコマンドを出す前に、データユニットインデックス部とALUを通してデータが流れるのを待たなければならないからである）。このステートマシンが使用される場合、ハフマンデコード自体がデータユニットインデックス部とALUにコマンドを発する。ハフマンデコードステートマシンはマイクロコード指令ビットの全てを制御することができないので、他のブロックに全範囲のコマンドを発することができない。

【1149】B. 2. 2. 1 「オペレーションの理論」

ハフマンコードを解読する時、本発明のハフマンデコードは入力コードをハフマンインデックス数に解読するために、演算手順を使用する。この数は（N個のエントリを持つコード表に対して）0～N-1の間である。ビットは入力シフトから1つずつ受け取られる。

【1150】マシンのオペレーションを制御するために、多くのテーブルが必要である。これらはコード（1～16ビット）において可能な各ビット数のために、その長さのコードがどれ程あるかを明記する。予期されるように、この情報は典型的に一般的なハフマンコードを明記するために充分ではない。しかしながら、MPEG、H. 261及びJPEGにおいて、ハフマンコードはこの情報だけでハフマンコード表を明記することができるように選択される。これには不運なことに1つだけ例外がある：MPEGにおいても使用されるH. 261

からのTcoefficientテーブルである。これは他の場所で説明される付加的なテーブルを必要とする（例外はスタートコードエミュレーションを避けるためにH. 261に計画的に導入された）。

【1151】このハフマンデコーダが使用するテーブルはJPEGに伝達されるものと正確に同一であることを認識することが重要である。これはハフマンデコーダの他のデザインが伝達されたものから内部テーブルの作成

```
int total=0;
int s=0;
int bit=0;
unsigned long code=0;
int index=0;
while (index>=total)
{
    if (bit>=max bits)
        fall (" huff decode: ran off end of
                huff table\n");
    code=(code<<1) Inext bit0;
    index=code-s+total;
    total+=codes per bit [bit];
    s=(s+codes per bit [bit])<<1;
    bit++;
}
```

一般に、特定の中間バリューはそれらが必要になる前にクロックフェイズで計算できるという事実に利点があるが、プロセスはシリコン実装の中に直接マッピングされる。

【1153】コードフラグメントから解くことは：

$$\text{EQ1. } \text{totaln}+1=\text{totaln}+\text{cpbn}$$

$$\text{EQ2. } ' \text{sn}+1=2 (' \text{sn}+\text{cpbn})$$

$$\text{EQ3. } \text{coden}+1=2 \text{ coden}+\text{bitn}$$

$$\text{EQ4. } \text{indexn}+1=2 \text{ coden}+\text{bitn}+\text{totaln}-' \text{sn}$$

運悪くハードウェアでは、「シフト済み」変数が変数“S”の代わりに使用される一連の修正された式を使用することがより簡単であることが解った。この場合：しかしながら、ハードウェアでは、「シフト済み」変数が変数“S”の代わりに使用される一連の修正された式を使用することがより簡単であることが解った。この場合：

$$\text{EQ5. } \text{shiftedn}+1=2 \text{ shiftedn}+\text{cpbn}$$

次のことが解る：

$$\text{EQ6. } \text{in}=2 \text{ shiftedn}$$

従って、これを式4に再び置き換えると、次の式が作られる：

$$\text{EQ7. } \text{indexn}+1=2 (\text{coden}-\text{shiftedn})+\text{totaln}+\text{bitn}$$

「インデックス」の連続する値を計算することに加え

を要求したかもしれない一方で、これらのテーブルが直接使用されることを可能にする。これは変換のために追加記憶装置及び追加プロセッシングを要求したであろう。MPEG及びH. 261のテーブルは（上述の例外はあるが）同じように説明でき、マルチスタンダードデコーダは実用的になる。

【1152】以下の“C”フラグメントはデコーディングプロセスを説明する：

て、計算がいつ完了するかを知ることが必要である。”
C”コードフラグメントから：

$$\text{EQ8. } \text{indexn}+1<\text{totaln}+1$$

の時に完了することが解る。

【1154】式7と式1から置き換えると、

$$\text{EQ9. } 2 (\text{coden}-\text{shiftedn})+\text{bitn}-\text{cpbn}<0$$

の時に完了することが解る。

【1155】本発明のハードウェア実装において、EQ. 7及びEQ. 9における共通の用語、(coden-shiftedn)は、これらの式の残りが評価されて最終結果、及び計算が「完了した」という情報を出す1フェイズ前に計算される。

【1156】警告ワードを1つ。様々な“C”コード、特に行動編集済みコードハフマンデコーダ、及びsm4 codeプロジェクトにおいて、“C”フラグメントはほぼ直接的に使用されるが、変数“S”は實際上「シフト済み」と称される。このように、「シフト済み」と呼ばれる2つの異なる変数がある。“C”コードの1つと、ハードウェア実装における他の1つである。これら2つの変数は2つの要因により異なる。

【1157】B. 2. 2. 1. 1 「データビットの反転」

ハフマンコードを正しく解読するために必要な情報がもう1つある。これは符号化データの極性である。H. 261とJPEGは反対の規定を使用することが解る。こ

れはH. 261のスタートコードはゼロビットであるのに対し、JPEGのマーカバイトは1ビットであるという事実から生じる。

【1158】両規定を処理するために、符号化データがH. 261スタイルのハフマンコードを解説するために、ハフマンデコードの中に読み込まれる時に、符号化データビットを反転させることが必要である。これは独占的ORゲートを用いる明白な方法で行われる。固定長コードを解説する時、データは反転されないの、反転はハフマンコードのためにのみ行われることに注目。

【1159】MPEGは2つの規定の混合物を使用する。H. 261から引き継がれる局面では、H. 261規定が使用される。JPEG (DCイントラ係数のデコーディング) から引き継がれる局面では、JPEG規定が使用される。

【1160】B. 2. 2. 1. 2 「変換係数表」
H. 261及びMPEGにおける変換係数表を使用する時、幾つかの変則がある。第1に、MPEGテーブルはH. 261テーブルのスーパーセットである。本発明のハードウェア実装において、2つのスタンダード間に引かれる区別はなく、これはテーブル (つまりMPEGコード) の拡張部分からのコードを含むH. 261ストリームが「正しい」方法で解説されるであろうということの意味する。もちろん、圧縮スタンダードの他の局面も同様に破壊される。例えば、これらの拡張コードはH. 261におけるスタートコードエミュレーションを生じさせるであろう。

【1161】第2に、変換係数表はcode per bitテーブルを用いる通常の方法では説明できないことを意味する変則を持つ。この変則は6ビット長のコードで発生する。これらのコードワードは交替コードワードで系統的に置き換えられる。エンコードにおいては、正しい結果が第1のエンコーディングによって通常の方法で得られる。次に、6ビットまたはそれ以上の長さの全てのコードのために、簡単なテーブルルックアップオペレーションにより、最初の6ビットが別の6ビットで置き換えられる。本発明によるデコードでは、デコーディングプロセスは6番目のビットが解説される直前に割り込まれ、コードワードはテーブルルックアップを用いて置き換えられ、デコーディングが続けられる。

【1162】この場合、10の可能な6ビットコードがあるだけであり、従って必要なルックアップテーブルは非常に小さい。更に、オペレーションはコードの上位2ビットがオペレーションによって変更されないという事実によって助けられる。その結果、真のルックアップテーブルを使用する必要がなくなる。その代わりに、小さなゲート集団が配線され、適切な変換が行われる。これを行うモジュールは“h f t c f r n g”と呼ばれる。可能性のあるコードセットからの各コードがそのセットからの別のコードで置き換えられる (新コードが導入さ

れない、あるいは旧コードが省略される) ので、このタイプのコード置換は本文では「リング」と規定される。

【1163】更に、ブロック内の最初の係数のために独特な実装が行われる。この場合、ブロック・エンドコードが発生することは不可能であり、従って、最も一般的に発生する記号が、そうでなければブロック・エンドとして解釈されるであろうコードを使用できるように、テーブルが修正される。これにより1ビットを節約できる。本発明によるデコーディング用の構成で、これは容易に収容される。簡単に言えば、「インデックス」がバリュウゼロであれば、第1係数の第1ビットのために、デコーディングは終了したとみなされる。更に、1ビットだけを解説後は、「インデックス」用には2つの可能性のあるバリュウ、ゼロと1だけがあり、1ビットをテストすることだけが必要である。

【1164】B. 2. 2. 1. 3 「レジスタ及びアダーのサイズ」

本発明のハフマンデコードは16ビットまでのハフマンコードを処理することができる。しかしながら、デコーディングマシンは8ビット幅である。これは解説されるハフマンインデックス数値が最大でも255であることが解っているから可能である。事実、これは拡張JPEGにおいてのみ発生でき、現在のアプリケーションでは、その限界は幾分低め (しかし、128より大きいので、7ビットでは不十分) である。

【1165】全ての合法的ハフマンコードに対して、「インデックス」の最終値だけでなく、全ての中間値も0~255の範囲にあることが解る。しかしながら非合法的コード、つまり (おそらくデータ誤差のため) 現在のコード表にはないコードを解説しようとする企てに対しては、インデックス値は255を越えることがある。我々は8ビットマシンを使用しているの、デコーディング・エンドで、「インデックス」の最終値が255を越えない。なぜなら、誤差が発生したことを我々に告げるより重要なビットが捨てられているからである。このため、デコーディング中のインデックス値が255を越える (つまり、インデックスを形成するアダーからのキャリー) 時にはいつでも、誤差が発生し、デコーディングが放棄される。

【1166】「コード」の12ビットが保護される。3ビットレジスタで充分であるハフマンコードを解説するためには、これは必要ではない。これらの上位ビットは12ビットまで読み出すことができる固定長コードのために必要である。

【1167】B. 2. 2. 1. 4 「固定長コード用のオペレーション」

固定長コードのために、「ビット毎のコード」バリュウがゼロになるように強いらられる。これはオペレーションを通して「全体の」と「シフト済み」がゼロのままであり、従って、「インデックス」はコードと同じである。

10

20

30

40

50

事実、アダー等は8ビットバリューだけが「インデックス」のために作られるようにする。このため、固定長コードを解説する時に、出力ワードの上位ビットは「コード」レジスタから直接取られる。ハフマンコードを解説する時、これら上位ビットはゼロになるように強いら

れる。
【1168】 充分なビットが入力から読み出されたという事実は明白な方法で計算される。コンパレータは所望のビット数を「ビット」カウンタと比較する。

【1169】 B. 2. 2. 2 「デコーディング係数データ」

本発明によるパーザーステートマシンは一般にかなり高レベルのデコーディングのためにのみ使用される。8ビットずつのデータブロック内の非常に低レベルのデコーディングは、直接にはこのステートマシンによって処理されない。パーザーステートマシンは「ブロックを解説する」形態のハフマンデコーダにコマンドを出す。ハフマンデコーダ、データユニットインデックス部、及びALUは（基本的にハフマンデコーダ内にあ

る）専用ステートマシンの制御下に、共に作動する。この配置はエントロピーコード化係数データの高性能のデコーディングを可能にする。更に本オペレーションモードにおいて操作される他のフィードバックパスがある。例えば、SIZE及びRUN情報を提供するためにVLCが解説されるJPEGデコーディングにおいては、SIZE情報はデータユニットインデックス部の出力から直接ハフマンデコーダに送り返され、どれほど多くのFLCビットを読むべきかをハフマンデコーダに指示する。それに加えて、幾つかの加速装置が実装される。例えば、同じ例を用いて、データインデックスステ

ージの前にハフマンインデックスバリューを考察することにより、ゼロのSIZEを生じる全てのVLCバリューが明白に捕獲される。これは非ゼロSIZEバリューの場合、ハフマンデコーダが実際のSIZEバリューが解る前に、1つのFLCビットを読むために進むことができることを意味する。これは最初のFLCビットの読出しがデータユニットインデックス部においてテーブルルックアップを遂行するために必要な1つのクロックサイクルに重なるので、如何なるクロックサイクルも浪費されないことを意味する。

【1170】 B. 2. 2. 2. 1 「MPEG及びH. 261のAC係数データ」
図133はAC係数がMPEG及びH. 261において解説される方法を示している。ハフマンデコーダのオペレーションを詳細に示すフローチャートが図128に示されている。

【1171】 VLCコードを読むことによりプロセスがスタートする。通常のイベントでは、ハフマンインデックスは6ビットRUN及び係数の絶対値を表すバリューに直接マッピングされる。その後、FLC1ビットが読

まれ、係数のサインを出す。ALUは係数の絶対値をこのサインビットで組み立て、係数の最終値を提供する。

【1172】 この時点でのデータフォーマットはサイン・マグニチュードであるので、このオペレーションにはほとんど困難なことがない。RUNバリューは6ビットの補助バスに送られる一方、係数バリュー（レベル）は通常のデータバスに送られる。

【1173】 2つの特別な場合があり、これらはデータインデックスオペレーションの前に解説されたインデックスバリューを考察することにより捕獲される。これらはブロック・エンド（EOB）とエスケープコード化データである。EOBの場合、これが発生したという事実はトークンフォーマッティング部が開かれたデータトークンを正しく閉じることができるよう、データユニットインデックス部及びALUブロックを通して送られる。

【1174】 エスケープコード化データはより複雑である。最初の6ビットのRUNが読み出され、これらはデータユニットインデックス部を通して直接送られ、ALUに記憶される。その後、FLC1ビットが読み出される。これはMPEG及びH. 261において説明されるエスケープの8ビットの内、最も重要なビットであり、レベルのサインを与える。ネガティブバリュー対ポジティブバリューのためにALUに異なるコマンドを送る必要があるため、そのサインは本実装において明白に読み出される。これはALUがビットストリーム内の2つの補数値をサインマグニチュードに変換できるようにする。いずれの場合にも、残りのFLC7ビットがその後読み出される。これがバリューゼロであれば、更に8ビットを読まなければならない。

【1175】 本発明では、ハフマンデコーダの内部ステートマシンは自身の制御のため、またデータユニットインデックス部、ALU、及びトークンフォーマッティング部をも制御するためにコマンドを発生させる責任がある。図133に示すように、ハフマンデコーダの指令は3つのソース；パーザーステートマシン、ハフマンステートマシン、もしくはパーザーステートマシンから以前に受け取ったレジスタに記憶されている指令の1つから送られる。本質的に、（ハフマンステートマシンに制御を引き継がせ、係数を読ませる）パーザーステートマシンからの原始指令は、レジスタに保持される、つまり、新しいVLCが必要になる度に、使用される。デコーディングのための他の全ての指令はハフマンステートマシンによって供給される。

【1176】 B. 2. 2. 2. 2 「MPEGのDC係数データ」

これはJPEGのDC係数データと同様に処理される。同じ（ロード可能な）テーブルが使用され、その内容が正しいことを保証することが制御マイクロプロセッサの責任である。MPEGスタンダードとの唯一の現実的な

差は、(J P E Gの場合と同様に) プレディクタがゼロに設定され、その補正が逆量子化器において行われることである。

【1177】B. 2. 2. 2. 3 「J P E G係数データ」

図129は本発明に従い、J P E GのAC係数を解読するためのハードウェアを示すブロック線図である。DC係数用のプロセスは本質的にJ P E Gプロセスを簡略化したものである。線図はAC及びDC係数両方のために作用する。M P E GのAC係数用の前の線図に現実に加えられる唯一のことは、" S S S S " フィールドが送り返され、読み出されるF L Cビット数を指定するため、ハフマンデコードコマンドの一部として使用できることである。残りのコマンドはハフマンステートマシンによって供給される。

【1178】図130はAC及びDC係数のハフマンデコーディング用のフローチャートを描いている。

【1179】AC係数用のプロセスをまず取り上げるが、プロセスは適切なテーブル(2つのACテーブルがある)を用いて、V L Cを読み出すことで開始する。ハフマンインデックスは次にデータユニットインデックス部内のR U N及びS I Z Eバリューに変換される。2つのバリューはハフマンインデックスステージで捕獲されるが、これらはE O B及びZ R L用である。これらは唯一如何なるF L Cビットも読まれない2つのバリューである。デコードインデックスがこれら2つのバリューのものではない場合、ハフマンデコードは直ちにF L C 1ビットを読み出す一方、データユニットインデックス部がルックアップオペレーションを完了して、実際にどの程度のビットが必要であるかを決定するのを待つ。E O Bの場合、ハフマンデコード内のハフマンステートマシンが遂行する更なるプロセッシングは必要ではなく、別のコマンドがパーザステートマシンから読み出される。Z R Lの場合、如何なるF L Cビットも必要ではないが、ブロックは完了されない。この場合、ハフマンデコードは(以前と同じテーブルを用いて)更なるV L Cの解読を直ちに開始する。

【1180】Z R L及びE O Bに連合するインデックスバリューを検出するのに、特別な問題がある。これはハフマンテーブルが(H. 261やM P E Gと異なり)ダウンロード可能であるからである。2個のJ P E GのACテーブルの各々のために、(1つはZ R L用に、1つはE O B用に) 2個のレジスタが提供される。これらはテーブルがダウンロードされる時にロードされる。それらは適切な記号と連合するインデックスバリューを保持する。

【1181】A L UはS I Z EビットのF L Cコードを適切なサイン・マグニチュードバリューに変換しなければならない。これらはテーブルがダウンロードされる時にロードされる。それらは適切な記号と連合するインデ

ックスバリューを保持する。A L UはS I Z EビットのF L Cコードを適切なサイン・マグニチュードバリューに変換しなければならない。これはまず間違ったサインでそのバリューをサイン拡張することにより行われる。サインビットが新たに設定されると、次に残りのビットが逆にされる(1の補数)。

【1182】DC係数の場合、Z R Lフィールドの等価物はないので、ハフマンデコーディングステージにおける意志決定の方が幾分容易である。ゼロのF L Cビットが読まれるようにする唯一の記号はゼロのDC差を示すものである。これは再びハフマンインデックスステージにおいて捕獲され、(ダウンロード可能な) J P E Gの各DCテーブル用にこのインデックスを保持するため、レジスタが提供される。

【1183】本発明のA L Uは(予測として知られる)最後のDC係数値のコピーを保持することにより、最終的な解読済みDC係数を形成するという仕事をする。4つの活性色成分の各々のために1つずつ、計4個のプレディクタが必要である。DC差が解読されると、A L Uは適切なプレディクタを加えて、解読された値を形成する。これはやはりその色成分の次のDC差のためのプレディクタとして記憶される。DC係数は(DCオフセットのため)サインされるので、2補数からサイン・マグニチュードへの変換が必要である。その後、そのバリューはゼロのR U Nと共に出力される。事実、このラストステージの一部を遂行せよという指令はハフマンステートマシンによって供給されない。それらは単にパーザステートマシンによって実行される。

【1184】AC係数に対する同様の方法において、A L UはまずF L CのS I Z EビットからDC差を生成しなければならない。しかしながら、この場合、2つの補数値がプレディクタに加算されることが求められる。これは前述のように、まず間違ったサインでサイン拡張することにより形成される。その結果が負であれば、正しい値を形成するために1を加算しなければならない。もちろん、これはアダーの中にキャリーを詰め込むことによりプレディクタと同時に加算され得る。

【1185】B. 2. 2. 3 「誤差処理」

誤差処理については説明する価値がある。検出される誤差には實際上4つのソースがある：

・テーブル・エンドから外れる。

【1186】・トークンが期待される時に、シリアルである。

【1187】・シリアルが期待される時に、トークンである。

【1188】・ブロック内に多くの係数がありすぎる。

【1189】これらの内最初のものは2つの状況において発生する。ビットカウンタが16に達する(合法的バリューは0~15である)場合、最長の合法的ハフマンコードが16ビットであるので、誤差が発生する。「イ

ンデックス」の中間値が255を越える場合、セクションB、2.2.1、3において説明したように、誤差が発生する。

【1190】第2のものはトークンが期待されるのにシリアルデータに遭遇する時に発生する。第3のものは反対の状態が生じる時に発生する。

【1191】最後のタイプの誤差は、ブロックにあまりに多くの係数がある場合に発生する。これは実際にはデータユニットインデックス部において検出される。

【1192】これらの状態のいずれかが発生すると、誤差はハフマン誤差レジスタにおいて注目され、パーザーステートマシンが割り込まれる。誤差を処理し、回復のために必要なコマンドを発することがパーザーステートマシンの責任である。

【1193】正しいオペレーションを保証するための割り込み時に、ハフマンはパーザーステートマシンと協働する。ハフマンデコードがパーザーステートマシンに割り込むと、新しいコマンドがパーザーステートマシンの出力においてアクセプトされるのを待つことができる。ハフマンデコードはパーザーステートマシンに割り込んだ後、2つの全サイクルのためにこのコマンドをアクセプトしないであろう。これにより、パーザーステートマシンは（今は実行されるべきではない）そこにあるコマンドを取り除き、適切なものと置き換えることができる。これら2つのサイクルの後、ハフマンデコードは通常のオペレーションを再び開始し、有効なコマンドがそこにあればそのコマンドを受け入れる。そうでなければ、パーザーステートマシンが有効なコマンドを示すまで何もしないであろう。

【1194】これらの誤差のいずれかが発生すれば、「ハフマン誤差」イベントビットが設定され、マスクビットが設定されれば、ブロックは停止し、制御マイクロプロセッサは通常の方法で割り込まれるであろう。

【1195】ある状況では、誤差のように見えるものが実際には誤差ではないことがある。このようなことが発生する最も重要な状況は、マクロブロックアドレスを読んでいる時である。MPEG、H.261及びJPEGのシンタックスにおいて、期待されるマクロブロックアドレスの代わりにトークンが発生するのは合法的である。合法的方法でこうしたことが発生すれば、ハフマン誤差レジスタにゼロがロードされる（誤差が無いことを意味する）が、パーザーステートマシンはまだ割り込まれたままである。パーザーステートマシンのコードはこのno errorの状況を確認し、それに従って応答する。この場合、「ハフマン誤差」イベントビット

は設定されず、ブロックはプロセッシングを停止しないであろう。

【1196】いろんな状況を処理しなければならない。まず、トークンは直ちに発生し、シリアルビットは前進しない。この場合、「シリアルが期待される時のトークン誤差」が発生したであろうが、その代わりに、no error誤差が前述したような方法で発生する。

【1197】第2に、トークンの前に2〜3のシリアルビットがある。この場合、決定が為される。トークンの前にある全てのビットがバリュース1を持っていた場合、（H.261及びMPEGでは、コード化データは反転されるので、コード化データファイルにはゼロビットがあることを思いだしてほしい）、ノーエラーが発生する。しかしながら、それらの内のどれかがゼロであれば、それらは有効なスタッフィングビットではなく、こうして誤差が発生し、「シリアルが期待される場合のトークン」誤差が発生する。

【1198】第3に、トークンの前には多くのビットがある。この場合、同じ決定が為される。全ての16ビットが1であれば、それらはパディングビットとして処理され、no error誤差が発生する。それらの内のいずれかがゼロであった場合、「ハフマンテーブルを外れる」誤差が発生する。

【1199】トークンが予期されずに発生する別の場所はJPEGである。ハフマンテーブルもしくは量子化器テーブルのいずれかを処理する場合、如何なる数のテーブルも同じマーカーセグメントの中に発生することができる。ハフマンデコードはいくつのテーブルがあるのかを知ることができない。このため、各テーブルが完了した後、ハフマンデコードは別のFLC4ビットを読み出し、それを新しいテーブルナンバーであると仮定する。しかしながら、新しいマーカーセグメントがスタートすると、4ビットFLCの代わりにトークンに遭遇するであろう。この要件は予測されず、従ってIgnore Errorsコマンドビットが加えられた。

【1200】B.2.2.4 「ハフマンコマンド」
ハフマンデコードブロック及びそれらの定義を制御するため、パーザーステートマシンが使用するビットをここに示す。データユニットインデックス部コマンドビットもこのテーブルに含まれる。マイクロプログラムの視点から、ハフマンデコード及びデータユニットインデックス部は1つの凝縮した論理ブロックとして作用する。

【1201】

【表175】

ビット	名称	機能
11	Ignore Errors	ある環境において誤差を不能化するために使用される。
10	Download	ダウンロード用のテーブルを指名するか、もしくはそのテーブルにデータをダウンロードする。
9	Alutab	テーブルナンバー (もしくはFLCビットナンバー) を指定するため、ALUレジスタからの情報を使用する。
8	Bypass	データユニットインデックス部にバイパスさせる。
7	Token	FLCもしくはVLCではなくトークンをデコードする。
6	First Coeff	Tcoeffテーブル及び他の特別なモードのために最初の係数トリックを選択する。
5	Special	ハフマンステートマシンを設定すれば、制御を引き継ぐべきである。
4	VLC (not FLC)	VLCまたはFLCを指定する。
3	Table [3]	VLCのために使用するためテーブルを指定する。
2	Table [2]	あるいはFLC用に読み出すビット数。
1	Table [1]	
0	Table [0]	

表B. 2. 2 ハフマンデコードコマンド

B. 2. 2. 4. 1 「FLC読出し」

このモードでは、Ignore Errors、Download、Alutab、Token、First Coeff、Special and VLCが全てゼロである。Bypassはデータインデックス変換が発生しないように設定される。

【1202】テーブル [3:0] の二進数は如何に多くのビットを読むべきかを指示する。ナンバー0~12は合法的である。バリューゼロは (期待されたであろうように) 実際にゼロビットを読み、従ってこの指令はハフマンデコードNOP指令である。バリュー13、14、15は作動せず、バリュー15はハフマンステートマシンが制御下にあり、読まれるべきFLCのビット数として"SSSS"の使用を表示する時に使用される。

30 【1203】B. 2. 2. 4. 2 「VLC読出し」

このモードでは、Ignore Errors、Download、Alutab、Token、First Coeff、and Specialがゼロであり、VLCは1である。通常、Bypassはデータインデックス変換が発生するように設定される。

【1204】このモードでは、Token、First Coefficient、及びSpecialは全てゼロであり、VLCは1である。

【1205】テーブル [3:0] の二進数は以下に示すように使用すべきテーブルを指示する。

【1206】

【表176】

テーブル [3:0]	使用するVLCテーブル
0000	TCoefficient (MPEG及びH. 261)
0001	CBP (コード化ブロックパターン)
0010	MBA (マクロブロックアドレス)
0011	MVD (モーションベクトルデータ)
0100	Intra Mtype
0101	予測Mtype
0110	補間Mtype
0111	H. 261 Mtype
10x0	JPEG (MPEG) DC Table 0
10x1	JPEG (MPEG) DC Table 1
11x0	JPEG AC Table 0
11x1	JPEG AC Table 1

表B. 2. 3 ハフマンテーブル

RAMに保持されるテーブル (つまりJPEGテーブル) の場合、ビット1は使用されず、テーブル選択が2回発生することに注目。非ベースラインJPEGデコーダが作られる場合、4個のDCテーブルと4個のACテーブルがあり、テーブル [1] が必要となる。

【1207】テーブル [3] がゼロであれば、テーブルがH. 261スタイルのテーブルとして正しく読まれるために、入力データは使用される時に反転される。テーブル [3:0] = 0の場合、適切なRing修正も適用される。

【1208】B. 2. 2. 4. 3 「NOP指令」
前述したように、ゼロビットのFLCを読み出す動作がNo Operation指令として使用される。入力ポートからは如何なるデータ (トークンまたはシリアル) のいずれも読み出されず、ハフマンデコーダは指令ワードと共にゼロのデータ値を出力する。

【1209】B. 2. 2. 4. 4 「TCoefficient第1係数」

H. 261及びMPEGのTCoefficientテーブルはブロックの1番目の係数のために使用される特別な非ハフマンコードを持つ。ブロック・スタートにおいてTCoefficientをデコードするために、

First CoefficientビットがVLC指令と共にテーブルゼロに設定されてもよい。First Coefficientビットの多くの効果の1つはこのコードがデコードされることを可能にすることである。

【1210】通常のオペレーションでは、TCoefficient VLCを読むために、「簡単な」コマンドを発することは異例であることに注目。これは通常Special Bitを設定することにより、制御がハフマンデコーダに手渡されるからである。

【1211】B. 2. 2. 4. 5 「トークンワードの読出し」

トークンワードを読むために、Tokenビットは1に設定すべきである。Special及びFirst Coefficientビットはゼロであるべきである。テーブル [0] ビットを正しく作用させる場合、VLCビットも設定されるべきである。

【1212】このモードでは、ビットテーブル [1] 及びテーブル [2] は以下のようにトークン読出し行為を修正するために使用される：

【1213】

【表177】

ビット	意味
Table [0]	シリアルデータのパディングビットを捨てる。
Table [1]	全てのシリアルデータを捨てる。

テーブル[0]及びテーブル[1]がゼロであれば、トークンの前のシリアルデータの存在はエラーであると考えられ、そのような信号が発せられる。

[1214] テーブル[1]が設定された場合、全てのシリアルデータはトークンワードに遭遇するまで捨てられる。このシリアルデータの存在故にエラーは発生しないであろう。

[1215] テーブル[0]が設定された場合、パディングビットは捨てられるであろう。もちろん、パディングビットの極性を知ることが必要である。これはVLCデータの読出しのための場合と全く同様の方法でテーブル[3]により決定される。テーブル[3]がゼロであるなら、入力データはまず反転され、それからどれかの「1」ビットが捨てられる。テーブル[3]が1に設定される場合、入力データは反転されず、「1」ビットが捨てられる。テーブル[3]ビットに応じてデータを反転するという動作はVLCビットでは条件付きであるので、このビットは1に設定されなければならない。パデ

ィングビットでないビットに遭遇する(つまり、H. 261及びMPEGにおける「1」ビットである)場合、誤差が報告される。

[1216] これらの指令では、1つのトークンワードだけが読まれることに注目。拡張ビットの状態は無視され、このビットをテストし、それに従って行動することはデマルチプレクサの責任である。マルチプルワードを読む指令も提供される - 特別指令に関するセクションを参照せよ。

[1217] B. 2. 2. 4. 6 「ALUレジスタはテーブルを指定する」

Alutabビットが設定されると、ALUレジスタファイル内のレジスタを使用して実際に使用されるテーブルナンバーを決定することができる。VLCビットと共にコマンドで供給されるテーブルナンバーはどのALUレジスタが使用されるかを決定する：

[1218]

[表178]

VLC	テーブル [3:0]	ALUテーブル
0	x0xx	fwd r size
0	x1xx	bwd r size
1	x0xx	dc huff [compid]
1	x1xx	ac huff [compid]

表B. 2. 4 ALUレジスタセクション

固定長コードの場合、ベクトルをデコーディングするために正しいビット数が読まれる。r sizeがゼロであれば、NOP指令が生じる。

[1219] ハフマンコードの場合、発生するテーブルナンバーは1に設定されたテーブル[3]であり、結果的に生じるナンバーはJPEGテーブルの1つを引用する。

B. 2. 2. 4. 7 「特別指令」

今まで説明してきた全ての指令(もしくはオペレーションモード)は「単純な」指令として考慮される。受け取られる各コマンドのために、(シリアルデータもしくはトークンデータのいずれであっても)適切な量の入力データが読まれ、結果として生じるデータが出力される。誤差が検出されなければ、正確に1つの出力がコマンド毎に発生するであろう。

[1220] 本発明では、特別指令は1つ以上の出力ワードが1つのコマンドのために発生することができることに特徴がある。この機能を果たすために、ハフマンコードの内部ステートマシンが制御を行い、パーザーが要求した指令が完了したと決定するまで、必要に応じてそれ自身指令を出すであろう。

[1221] 全ての特別指令において、実行されることになるシーケンスの最初の実際の指令は1に設定される

Specialビットで出される。これは全てのシーケンスが独得の最初の指令を持たねばならないことを意味する。この企ての利点は、パーザーから受け取るコマンドに基づいて要求されるルックアップオペレーションなしに、シーケンスの最初の実際の指令が利用できることである。

[1222] 4つの認識される特別指令がある：

- ・TCoefficient
- ・JPEG DC
- ・JPEG AC
- ・Token

第1のものはブロック・エンド記号が読まれるまで、H. 261及びMPEG変換係数等を読む。ブロックが非イントラブロックである場合、このコマンドは全ブロックを読むであろう。この場合、First Coefficientビットは第1の係数トリックが適用されるように設定されるべきである。ブロックがイントラブロックであれば、DCタームが既に読まれているべきであり、First Coefficientビットはゼロであるべきである。

[1223] H. 261のイントラブロックの場合、DCタームはFLC8ビットバリューを読むために「単純な」指令を用いて読まれる。MPEGでは、下記におい

30

40

50

て説明するJ P E G D Cの特別指令が使用される。J P E G D Cコマンドは(V L Cにより示されるF L CのS S S Sビットを含む)J P E GスタイルのD Cタームを読むために使用される。F i r s t C o e f f i c i e n tビットはデータユニットインデックス部において(係数の数をカウントする)カウンタがリセットされるために設定されなければならない。J P E G A CコマンドはD Cタームの後、E O Bに遭遇するか、もしくは64番目の係数が読まれるまで、ブロックの残りを読むために使用される。

【1224】Tokenコマンドは全トークンを読むために使用される。トークンワードは拡張ビットが明白になるまで読まれる。それは未認識のトークンを処理する便利な方法である。

テーブル [3:0]	指名されるテーブル
10xx	ビット毎のJ P E G D Cコード
11xx	ビット毎のJ P E G A Cコード
00xx	J P E G D Cデータインデックス
01xx	J P E G A Cデータインデックス

表B. 2. 5 J P E Gテーブル

上記の表が示すように、A CもしくはD Cテーブルのいずれかをロードすることができ、テーブル[3]はそれが(ハフマンデコード自体における)codes-per-bitテーブルであるか、ロードされるデータインデックステーブルであるかを決定する。

【1227】テーブルが一旦指名されると、Downloadビットセット(及びF i r s t C o e f fビットゼロ)で、必要な数の(常に8ビットである)F L Cを読むためのコマンドを出すことにより、データはそのテーブルの中にダウンロードされる。これは符号化データが指名されたテーブルの中に書き込まれるようにする。アドレスカウンタは維持され、データは現在のアドレスに書き込まれ、次にアドレスカウンタが増加される。アドレスカウンタはテーブルが指名される時いつでもゼロにリセットされる。

【1228】データインデックステーブルをダウンロードする時は、データ及びアドレスが監視される。アドレスはハフマンインデックスナンバーである一方、そのアドレスにロードされるデータは最終的に解読される記号であることに注目。この情報は関係のある記号のためにハフマンインデックスナンバーを保持するレジスタを自動的にロードするために使用される。従って、J P E G A Cテーブルにおいて、Z R Lに対応するバリューを持つデータが認識される時、現在のアドレスはテーブルナンバーによって指示されるように、レジスタC E D H KEY Z R L INDEX 0またはC E D H

【1225】B. 2. 2. 4. 8 「ダウンロードテーブル」

本発明では、ハフマンデコードテーブルはDownloadビットを用いてダウンロードすることができる。最初のステップはダウンロードすべきテーブルを指名することである。これはDownloadビットセットとF i r s t C o e f fビットセットで、F L Cを読むためのコマンドを出すことによって行われる。これはN O Pとして処理され、従って如何なるビットも実際には読まれないが、テーブルナンバーはレジスタに記憶され、次のダウンロードにおいてどのテーブルがロードされるかを特定するために使用される。

【1226】

【表179】

KEY Z R L INDEX 1に書き込まれる。

【1229】解読されたデータはそれが解読された1フェイズ後に、codes-per-bitテーブルに書き込まれるので、このフェイズ中にテーブルからデータを読むことはできない。従って、テーブルダウンロード指令の後、直に出されるV L Cを読むとする指令は失敗するであろう。実際のアプリケーションにおいて(つまり、J P E Gを行う時)、該かるシーケンスが発生する理由はない。しかしながら、これを行うシミュレーションテストを構築することは可能である。

【1230】B. 2. 2. 5 「ハフマンステートマシン」

本発明によるハフマンステートマシンはある場合に内部的に作られるハフマンデコードコマンドを提供するために作用する。内部ステートマシンによって作られ得る全てのコマンドはデマルチプレクサによりハフマンデコードに提供され得る。

【1231】ステートマシンの基本構造は次のようである。コマンドがハフマンデコードに出されると、それは一連の補助ラッチに記憶され、後に再使用することができる。更にコマンドはハフマンデコードによって実行され、ハフマンステートマシンによって分析される。コマンドが公知の指令シーケンスの最初のものであると認識され、S P E C I A Lビットが設定されると、ハフマンデコードステートマシンはパーザステートマシンからハフマンデコードの制御を引き継ぐ。この時点

で、ハフマンデコード用の指令には3つのソースがある。

1) パーザーステートマシン：このチョイスは特別指令の完了時（例えば、EOBが解読された時）に行われ、次のデマルチプレクサコマンドがアクセプトされる。

【1232】2) ハフマンステートマシン。ハフマンステートマシンは任意のコマンドを持つことができる。

【1233】3) パーザーステートマシンによって、10 指令をスタートするために出される原始指令。

【1234】(2) の場合、テーブルナンバーはデータユニットインデックス部からのフィードバックによって提供されることが可能であり、これはハフマンステートマシンROM内のフィールドを取り替えるであろう。

【1235】(1) の場合、ある場合には、テーブルナンバーは（例えば、AC及びDCテーブルナンバー及びF-ナンバーの場合）ALUレジスタファイルから得られるバリューによって提供される。これらのバリューは補助コマンド記憶装置に記憶され、そのコマンドが後に20 再使用される時、テーブルナンバーは記憶されたものである。カウンタは通常次のブロックを引用するために進んでいるので、それはALUから再びリカバーされない。

【1236】使用される次の指令のチョイスは解読されつつあるデータ次第であるので、決定はサイクルの最後の方で行われることが必要である。従って、一般的な構造は可能な指令の全てが並列に準備され、サイクル後半のマルチプレクシングが実際の指令を決定するようになっている。

【1237】各々の場合に、次のサイクルの中でハフマンデコードが使用するであろう指令を決定することに加え、ステートマシンROMも、現在のデータがデータユニットインデックス部とそれからALUへと通過する時に、現在のデータに付けられるであろう指令を決定することに注目。全く同じ方法で、これら3つの指令全てが並列に準備され、チョイスがサイクル後半で行われる。

【1238】ここでやはり、上述のように次のハフマンデコード指令のための3つのチョイスに対応する、指令のこの部分のために3つのチョイスがある。40

【1239】1) ブロック・エンドにふさわしい一定の指令。

【1240】2) ハフマンステートマシン。ハフマンステートマシンはデータユニットインデックス部のために任意の指令を提供することができる。

【1241】3) 指令をスタートするためにパーザーが出す原始指令。

【1242】B. 2. 2. 5. 1 「EOBコンパレータ」

EOBコンパレータの出力は本質的に定数指令のセレクションがデータユニットインデックス部に表示されるように強制し、次のハフマン指令がパーザーからの次の指令であるようにさせるであろう。コンパレータの正確な機能はハフマンステートマシンROM内のビットによって制御される。

【1243】EOBコンパレータの背後に、AC及びDCのJPEGテーブルにEOB記号のインデックスを保持する4個のレジスタがある。DCテーブルの場合、もちろんブロック・エンド記号はないが、ゼロサイズの記号があり、それはDCゼロ差によって作られる。これはEOB記号の場合と全く同様にFLCのゼロビットが読まれるようにするので、それらは全く同様に処理される。

【1244】レジスタに保持される3つのインデックスバリューに加えて、定数値1も使用できる。これはH. 261及びMPEGにおけるEOB記号のインデックスナンバーである。

【1245】B. 2. 2. 5. 2 「ZRLコンパレータ」

本発明では、これは汎用コンパレータである。それはハフマンステートマシン指令もしくはI to Dによって使用されるためのオリジナル指令のいずれかを選択させる。

【1246】ZRLコンパレータの背後に、4つのバリューがある。2つはレジスタの中にあり、ZRLコードのインデックスをACテーブルの中に保持する。他の2つのバリューは定数であり、1つはバリューゼロであり、他の1つは12（MPEG及びH. 261におけるESCAPEのインデックス）である。定数ゼロはFLCの場合に使用される。定数12はテーブルナンバーが8より小さい時（及びVLC）はいつでも使用される。2つのレジスタの1つはテーブルナンバーの低オーダービットにより判断されて、テーブルナンバーが7より大きい場合（及びVLC）に使用される。

【1247】ステートマシンROM内のビットはコンパレータを可能化するために提供され、また別のビットがその動作を反転するために提供される。指令の中のT OKENビットが設定された場合、コンパレータ出力は無視され、拡張ビットによって置き換えられる。これはトークンの終わりまで続けられる。

【1248】B. 2. 2. 5. 3 「ハフマンステートマシンROM」

ハフマンステートマシン内の指令フィールドは次の通りである：

next state [4:0]

次のサイクルにおいて使用するアドレスである。このアドレスは修正することができる。

statectl

50 次のステートアドレスの修正を可能にする。ゼロであれ

ば、ステートマシンアドレスは修正されず、そうでなければ、アドレスのLSBが次の2つのコンパレータのいずれかのバリューで置き換えられる：

[1249]
[表180]

nextstate [0]	
0	EOB適合でLsbを置き換える
1	ZRL適合でLsbを置き換える

注：いずれの場合にも、次のハフマン指令が「リラン原始コマンド」として選択されると、ステートマシンはコマンドのために適切なものとしてロケーション0、1、2または3にジャンプするであろう。
eobct [1:0]

これは次のようなEOBコンパレータ及びextnビットに基づいて次のハフマン指令のセレクションを制御する：
[1250]
[表181]

eobct [1:0]	
00	効果なし、zrlct [1:0] を参照
01	EOBであれば、新しい (PARSER) コマンド
10	extnビットであれば、新しい (PARSER) コマンド
11	無条件のデマルチプレクサ指令

zrlct [1:0]

これはZRLコンパレータに基づいて次のハフマン指令のセレクションを制御する。条件が合えば、ステートマシン指令を取り、そうでなければ原始指令をリランする。いずれの場合にも、eobct 1+条件がデマルチ

プレクサ指令を取れば、これ (eobct 1+) は次のように優先権を持つ：
[1251]
[表182]

zrlct [1:0]	
00	決してSMを取らない (常にリラン)
01	常にSMコマンドを取る
10	ZRLが適合すればSM
11	ZRLが適合しなければSM

smtab [3:0]

本発明では、選択された指令がステートマシン指令であれば、これはハフマンデコードによって使用されるテーブルナンバーである。しかしながら、ZRLコンパレータマシンであれば、zrltab [3:0] フィールドが優先して使用される。

[1252] ZRL適合が発生するか否かに応じて、別のテーブルナンバーが使用される必要がない場合、smtab [3:0] 及びzrltab [3:0] は同じバリューを持つであろう。しかしながら、これはLsimにおいて奇妙なシミュレーション問題を導く可能性があることに注目。MPEGの場合、ZRLのためにハフマンインデックスナンバーを指示するレジスタをロードする明らかな必要はない (JPEGのみの構成)。しかし

ながら、これらは今も選択され、ZRLコンパレータが「未知」であるかもしれない (従ってどちらが選択されようと重大ではない) 全ての場合に、smtab [3:0] 及びzrltab [3:0] は同じバリューを持つという事実にもかかわらず、ZRLコンパレータは「未知」となり、次のステートも「未知」になるであろう。
zrltab [3:0]

これは選択された指令がステートマシン指令であれば、ハフマンデコードによって使用されるであろうテーブルナンバーである。しかしながら、ZRLコンパレータが適合すれば、zrltab [3:0] フィールドは優先して使用される。

[1253] ZRL適合が発生するか否かに応じて、別のテーブルナンバーが使用される必要がない場合、sm

tap[3:0]及びzrltab[3:0]は同じバリューを持つであろう。しかしながら、これはlsimにおいて奇妙なシミュレーション問題を導く可能性があることに注目。MPEGの場合、ZRLのためにハフマンインデックスナンバーを指示するレジスタをロードする明らかな必要はない(JPEGのみの構成)。しかしながら、これらは今も選択され、ZRLコンパレータが「未知」であるかもしれない(従ってどちらが選択されようとして重大ではない) 全ての場合に、smtab[3:0]及びzrltab[3:0]は同じバリューを持つ 10 という事実にもかかわらず、ZRLコンパレータは「未知」となり、次のステートも「未知」になるであろう。zrltab[3:0]これは選択された指令がステートマシン指令であれ

は、ハフマンデコードによって使用されるであろうデューブルナンバーであり、ZRLコンパレータが適合する。smv|c

これは選択された指令がステートマシン指令であれば、ハフマンデコードによって使用されるVLCビットである。

aluzrl[1:0]

このフィールドはALUに送られる指令のセレクションを制御する。それは(指令シーケンスのスタートで記憶された)パーザステートマシンからのコマンドであっても、あるいはステートマシンからのコマンドであってもよい:

[1254]

[表183]

aluzrl[1:0]	
00	常に置えられたParserステートマシンコマンドを取る
01	常にハフマンステートマシンコマンドを取る
10	EOBでなければハフマンSMコマンドを取る
11	ZRLでなければハフマンSMコマンドを取る

alueob

この配線はEOBコンパレータに基づいてALUに送られる指令の修正を制御する。これは単にALU出力モードをzinputに強制する。これは任意のチョイスである; noneから離れた出力モードであれば何でも良

い。これはブロック・エンドコマンドワードがトークンフォーマッティング部に送られ、そこでそれはデータトークンの適切なフォーマッティングを制御する:

[1255]

[表184]

alueob	
0	ALUのoutstcフィールドを修正しない
1	EOB適合の場合、"zinput"をoutstcに強制する

残りのフィールドはALU指令フィールドである。これらはALUの説明において適切に文書化される。

[1256] B. 2. 2. 5. 4 「ハフマンステートマシンの修正」

ステートマシンの1つの態様では、データユニットインデックス部は、エスケープコード化されたTcoef 40 efficientのRUN部分がいつデータユニットインデックス部に送られるかを「知る」必要がある。これは制御ROM内の適切なビットを用いて達成できる一方、ROMの変更を避けるため、別のアプローチが使用されてきた。この点に関して、ROMに入るアドレスは監視され、アドレスバリュー5が検出される。これはRUN

フィールドを処理するROMにおいて指名される適切な場所である。もちろん、他の被選択アドレスバリューを使用するようにROMをプログラムしようと思えばできたことは自明であろう。更に、制御ROMのビットを使用する前述のアプローチも利用できたであろう。

[1257] B. 2. 2. 6 「概略ガイドツアー」

本発明では、ハフマンデコードはhdと呼ばれ、hdは実際にデータユニットインデックス部を含む(これは編集されたコード生成の制限により必要とされる)。従って、hdは次の主要ブロックを含む;

[1258]

[表185]

モジュール名	説明
hddp	ハフマンデコーダ(算数) データバス
hdstdp	ハフマンステートマシンデータバス
hfitod	データユニットインデックス部

表B. 2. 6 ハフマンモジュール

ハフマンモジュールの以下の説明は、当業者であれば容易に理解できる、図面の詳細な部分において示した様々なサブシステムエリアの全体的な説明により行われる。

【1259】B. 2. 2. 6. 1 「“hd”の説明」
2線式インターフェース制御用のロジックは通常2線式インターフェースにより制御される3ポート：データ入力、データ出力、及びコマンドを含む。それに加えて、入力シフトからの2つの「有効な」線がある：トークンがin_data[7:0]に表示されることを指示するtoken_validと、データがシリアル上に表示されることを指示するserial_validである。

【1260】発生される最も重要な信号はラッチに行くイネーブルである。最も重要なものはph1ラッチに対するイネーブルであるe1である。ph0ラッチの多くは可能化されないが、2つのイネーブルがこれらのために提供され、それらはシリアルデータと関連するe0とトークンデータと関連するe0tである。

【1261】本発明では、done信号(done、notdone、及びそれらのph0バリエーションのdone0とnotdone0)は、原始ハフマンコマンドが完了する時を指示する。ハフマンステートマシンコマンドが実行される場合、doneは全ステートマシンコマンドを含む各原始コマンドの完了時に認定されるであろう。信号notnewは、全ハフマンステートマシンコマンドが完了するまで、パーザステートマシンからの新しいコマンドのアクセプタンスを防止する。

【1262】データユニットインデックス部から受け取る情報の制御に関して、sizeフィールド用の制御ロジックはJPEG係数デコーディング中にハフマンデコーダに送り返される。これは実際には2つの方法で起こる。サイズが正確に1であれば、これは専用信号notfbone0に送り返される。そうでなければ、データユニットインデックス部の出力から送り返される。(out_data[3:0]及び信号fbvalid1はこれが発生するのを指示する。フィードバックデータのコマンドレジスタ(シート10)へのマルチプレクシングを制御するために信号muxsizeが生み出される。)

それに加えて、正確に64の係数がデコードされたフィードバックがある。JPEGにおいてEOBはこの場合にコード化されないため、信号forceeobが作られる。類似により、上述したように、フィードバックサ

イズ用の信号と共に、これが実施されるには実際2つの方法がある。通常のフィードバックが行われる(jpeg_eob)場合、データがフィードバックされるにつれて、ラッチi-971だけがロードされ、新しいパーザステートマシンコマンドがアクセプトされるまでクリアされない。信号forceeobはハフマンコードがデコードされるまで実際に作られない。このように、固定長コード(つまり、サイズビット)は影響を受けないが、次のハフマンコード化情報は強制されたブロック・エンドにより置き換えられる。サイズが1であり、jpeg_eob0が使用された場合、1ビットだけが読み出されるので、i-1255とi-1256は正確な時間まで信号を遅らせる。注目すべきことは、ゼロサイズを持つ唯一の記号はEOBとZRLであるので、この状況においてゼロサイズが発生することが不可能であるということである。

【1263】デコーディングはtcoeff_tab0(Tcoeffテーブルを用いるハフマンデコーディング)、mba_tab0(MBAテーブルを用いるハフマンデコーディング)、及びnop(ノーオペレーション)を作るためのコマンドのかなりランダムなデコーディングである。nopを発生させるにはいくつかの理由がある。サイズゼロの固定長コードはその一つであり、forceeob信号もその一つであり(EOBを合図するために出力が作られても、入力シフトから如何なるデータも読み出されるべきではないので)、最後にテーブルダウンロード指名が第三の理由である。

【1264】(サイズゼロのFLC、NOPにより作られる)notfrczeroは、NOP指令が使用される時に、その結果がゼロであることを保証する。更に、invertはハフマンデコーディングの前にシリアルビットが反転されるべきであることを指示する(セクションB. 2. 2. 1. 1を参照)。ringは変換係数ringが適用されるべき時を指示する(セクションB. 2. 2. 1. 2を参照)。

【1265】デコーディングは更にcodes-per-bit ROMのアドレスに関して達成される。これらは小さなデータバスROMから構築される。信号は二重(例えば、cshaとcs1a)にされる。アドレスは選択されるブロックへのUPIアクセスに応じて、ビットカウンタ(bit[3:0])もしくはマイクロプロセッサインターフェースアドレス(key_addr[3:0])から取ることができる。

10

20

30

40

50

【1266】追加デコーディングはJPEGテーブル(EOB、ZRL等)用のハフマンインデックスバリュを保持するもののようなレジスタのUPI読み出しに関する。更に含まれるのは、これらのレジスタ及びcodes-per-bit RAMのUPI読み出し用のトライステートドライバ制御である。

【1267】演算データバスデコーディングも特定の重要なビット数のために提供される。first bitはTcoeffの第1の係数トリックに関連して使用され、bit fiveはTcoeffテーブルの中にリ50ングを適用することに関係する。EOBコンパレータがデコードされたインデックスバリュに適合する動作をシミュレートするためのforce_eobの使用に注目。

【1268】extnビットに関して、トークンが入力シフトから読まれる場合、関連extnビットはそれと共に読まれる。そうでなければ、extnの最後のバリュは保存される。これはトークンが読まれた後いつでも、マイクロコードプログラムによるextnビットのテストを可能にする。

【1269】zerodatが認定される時、ハフマン出力データの上位4ビットがゼロになるように強いられる。これらは固定長コードをデコーディングする時に有効なバリュを持つだけなので、それらはVLC、トークンをデコーディングする時、あるいはNOP指令が何等かの理由で実施される時はゼロにされる。

【1270】更に、回路は各コマンドが完了する時を検出し、done信号を発する。本質的に、doneである理由には2つのグループがある；通常の原因と、例外的な原因である。これらは各々2つの三路マルチプレクサの1つによって処理される。

【1271】下位マルチプレクサ(i-1275)が通常の原因を処理する。FLCの場合、信号ndnflcが使用される。これはビットカウンタをテーブルナンバーと比較するコンパレータの出力である。VLCの場合、信号ndnvlcが使用される。これは演算データバスからの出力であり、式9を直接的に反映する。NOP指令またはトークンの場合、1つのサイクルだけが必要であり、従って、システムは無条件にdoneである。

【1272】本発明では、上位マルチプレクサ(i-1274)が例外的な場合を処理する。JPEGデコーディングにおいてデコーダがサイズのフィードバックを期待している(fbexpctd0)場合、1つだけのビットが必要なのでデコーダはdoneである。デコーダがTcoeffテーブルを用いて最初の係数の第1ビットを処理している場合、現在のインデックスのビットゼロがゼロである場合、それはdoneである(セクションB. 2. 2. 1. 2を参照)。これらの条件のいずれも満たされない場合、doneであるための例外的な理50

由はない。

【1273】NORゲート(i-1293)は最終的にdone状態を変形させる。i-570(つまり、データが有効ではない)により生じる状態がdoneを強いる。これは少しおかしく思われるかもしれない。それは基本的に、第1のコマンドの(doneが全てのカウンタ、レジスタ等をリセットする)ための準備のために、マシンをdone状態にならせるためにリセット後使用される。

【1274】信号notdonexは誤差を検出する際に必要である。誤差を検出すれば、doneがとにかく強いられるので、通常のdone信号は使用できない。doneの使用は結合フィードバックループを与えるであろう。

【1275】誤差検出及び処理は可能性のある全ての誤差状態を検出する回路によって行われる。これらはi-1190において共に0Redである。この場合、i-1193、i-585及びi-584は3ビットのハフマン誤差レジスタを構成する。「実際の」誤差がない場合に誤差を不能化するi-1253及びi-1254に注目(セクション B. 2. 2. 3を参照)。

【1276】それに加えて、i-580及びi-579は関連する回路と共に、誤差検出後最初のコマンドのアクセプタンスを制御する単純なステートマシンを提供する。

【1277】前述したように、制御信号はデータユニットインデックス部及びALUにおいてパイプライン遅延に合わせるため遅延される。

【1278】Iotd_bypassはデータユニットインデックス部に送られる実際のバイパス信号である。それは固定長コードがデコードされる時はいつでも、ハフマンステートマシンがバイパスしなければならないように制御される時に修正される。

【1279】Aluinstr[32]はALUがパーザステートマシンに(コンディショニングコードを)フィードバックするようにさせるビットである。更に、ハフマンステートマシンが制御されている時、信号は(原始コマンドの1つが完了する都度為されるよりもむしろ)一度だけ認定されることが重要である。

【1280】Aluinstr[36]は(他のALU指令ビットが増分を明記する場合)ALUがブロックカウンタのステップを踏むようにさせるビットである。これも一度だけ認定されなければならない。

【1281】それに加えて、これらのビットはトークン変換にデータを出力するALU指令のためにのみ認定されねばならない。そうでなければ、カウンタはトークン変換への最初の出力前に増分され、データトークンにおいてccという不正確なバリュを生じるかもしれない。

【1282】本発明の図示した態様では、ALUがトー

クンフォーマッティング部に出力される場合、`alunode[1]` もしくは `alunode[0]` のいずれかが低くなるであろう。

【1283】図127は `hdstdp` と称されるハフマンステートマシンデータバスを示す。更にハフマンステートマシンROMの出力を読むためのUPIデコードがある。

【1284】ロケーションの場合に処理するためマルチプレクシングが提供される（セクションB2.2.4.6を参照）。

【1285】`aluinstr[3:2]` の修正は `ALUoutsrsc` 指令フィールドを `non-none` に強いる処理をする（セクションB.2.2.5.3、`alueob` の説明を参照）。

【1286】ハフマンデコードブロック (x) のためのコマンドレジスタに関して、コマンドの各ビットはコマンドの可能なソース間の選択をする関連マルチプレクサを有する。4つの制御信号はこのセクションを制御する：`Selhold` はレジスタに現在のステートを保持するようにさせる。

【1287】`Selnew` はパーザステートマシンから新しいコマンドをロードさせる。これは更に後の使用のために、元のパーザステートマシンのコマンドを保持するレジスタのローディングをも可能にする。

【1288】`Selold` は元のパーザステートマシンのコマンドを保持するレジスタからコマンドのローディングを起こさせる。

【1289】`selism` はハフマンステートマシンROMからのコマンドのローディングを起こさせる。

【1290】テーブルナンバーの場合、テーブルナンバーがデータユニットインデックス部の出力データからもロードされるので、状況はそれより少し複雑である（`selholdt` 及び `muxsize`）。

【1291】ラッチはハフマンステートマシンROMの中に現在のアドレスを保持する。ロジックは可能な4つのコマンドのどれが実行されているかを検出する。これらの信号は新しいコマンドの場合スタートアドレスの下位2ビットを形成するために組み合わせられる。

【1292】ロジックは更にステートマシンROMの出力が無意味である（通常はコマンドが「単純な」コマンドであるため）時を検出する。信号 `notignorerom` はステートマシンのオペレーションを効果的に不能化し、特にALUに送られる指令の修正を不能化する。

【1293】`fixstate0` を生じさせる回路はこのステートマシンの制限されたジャンピング能力を制御する。

【1294】デコーディングは更にハフマンステートマシンROMへと信号を追いやるために提供される。これはデータバススタイルの結合ROMである。

【1295】`escape run` の発生はセクションB2.2.5.4において説明される。

【1296】更に、デコーディングはZRLやEOB等の記号のためにハフマンインデックスナンバーを保持するレジスタの準備をする。これらのレジスタはUPIまたはデータバスからロードすることができる。`center(es[4:0]` 及び `zs[3:0]`) におけるデコーディングはデコードハフマンインデックスと比較するためにどのレジスタまたは一定バリューかを選択するマルチプレクサのためにセレクト信号を発生させている。

【1297】ハフマンステートマシン用の制御ロジックに関して。ここでは、ハフマンステートマシンROMからの「指令」ビットは、次に何をすべきか、またALU用の指令ワードを如何に修正するかを決定するため、様々な条件と組み合わせられる。

【1298】本発明では、信号 `notnew` 及び `notold` はハフマンデコードコマンドレジスタのオペレーションを制御するため、シート10上で使用される。それらはここではハフマンインデックスコンパレータ (`neobmatch` 及び `nzrlmatch`) と共に、ステートマシンROM（セクション B.2.2.5.3において説明される）の中の制御ビットから明らかな方法で作られる。

【1299】更に、ALUに送られる指令用のソースのセクションが行われる。実際のマルチプレクシングはハフマンステートマシンデータバス `hfstdp` において行われる。4つの制御信号が発生される。

【1300】ブロック・エンドに遭遇しなかった場合、`aluselmx`（パーザステートマシン指令を選択する）もしくは `aluselism`（ハフマンステートマシン指令を選択する）のいずれか1つが作られるであろう。それに加えて、ALU指令の `outsrsc` フィールドがそれを `zinput` に押しやるために修正される。

【1301】レジスタはテーブルダウンロード中に指名されたテーブルナンバーを保持する。デコーディングは `codes-per-bit` RAMのために提供される。追加デコーディングは、ハフマンインデックスナンバーレジスタが自動的にロードされるように、EOM及びZRL等の記号がダウンロードされる時を認識する。

【1302】ビットカウンタに関して、コンパレータはFLCを読む時に、いつ正しい数のビットが読まれているかを検出する。

B.2.2.6.2 「“hddp”の説明」

コンパレータはハフマンインデックスの特別なバリューを検出する。レジスタはダウンロード可能なテーブル用のバリューを保持する。マルチプレクサ (`meob`

`[7:0]` 及び `mzr[7:0]`) は使用すべきバリューを選択し、`exclusive-or` ゲート及びゲーティングがコンパレータを構成する。

【1303】アダー及びレジスタはセクション B. 2. 2. 1において説明した式を直接評価する。更なる説明はここでは不要であると考えられる。排他的論理和はセクション B. 2. 2. 1. 1において説明したデータ (i-807) を反転するために使用される。

【1304】codeレジスタは12ビット幅である。マルチプレクシングアレンジメントはセクション B. 2. 2. 1. 2において説明した ring 置換を履行する。

【1305】データ用のパイプライン遅延及びデコード 10 化シリアルデータ (index [7:0]) とトークンデータ (ntoken0 [7:0]) 間のマルチプレクシングに関して、ハフマンインデックスバリュースは ZRL 及び EOB 記号の中で決定される。

【1306】Codes-per-bit ROM 及びそれらのマルチプレクシングは使用すべきテーブルを決定するために使用される。テーブルセレクト情報が遅れて到着するので、このアレンジメントが使用される。その後全てのテーブルがアクセスされ、正しいテーブルが選択される。

【1307】Codes-per-bit RAM に関して、codes-per-bit ROM の最終マルチプレクシング及び codes-per-bit RAM の出力はブロック hdcpram の内部で起こる。

【1308】B. 2. 2. 6. 3 「“hdstdp” の説明」

本発明において、Hdstdp は2つのモジュールから成る。hdstdel は適切なパイプラインステージまで、例えば、それらが ALU 及びトークン変換に供給される時まで、パーザーステートマシン制御ビットを遅 30 らせることに関係がある。それは ALU に送られる指令

ワードの約半分だけを処理し、残りは他のモジュール hdstdmod により処理される。

【1309】Hdstdmod はハフマンステートマシン ROM を含む。この指令の数ビットはハフマンステートマシン制御ロジックにより使用される。残りのビットは (パーザーステートマシンからの) ALU 指令ワードの hdstdel において処理されない部分を置き換えるために使用される。

【1310】Hdstdmod は自明であり、何の説明も必要ではない。パイプラインディレイレジスタだけがある。

【1311】Hdstdel も非常に単純であり、ROM 及び ALU 指令を修正するマルチプレクサにより処理される。回路の残部はハフマンステートマシン ROM 出力の半分に対する UPI 読み取りアクセスに関係する。バッファも制御信号のために使用される。

【1312】B. 2. 3 「トークンフォーマッティング」

本発明によるハフマンデコードトークンフォーマッティングは、ハフマンブロックの終了部にある。その機能は、その名前が示す通り、ハフマンデコードからのデータを適当なトークン構造にフォーマットすることである。入力データはマイクロインストラクションワードコマンドフィールドの制御下、マイクロインストラクションワードの中のデータで多重送信される。ブロックは2つの操作モード; データワードとデータトークンを持っている B. 2. 3. 1 「マイクロインストラクションワード」

【1313】

【表186】

フィールド名	ビット
Token	0:7
Mask	8:11
Block Type (Bt)	12:13
External Extn (Ee)	14
Demux Extn (De)	15
End of Block (Eb)	16
Command (Cmd)	17

17	16	15	14	12	8	0
Cmd	Eb	De	Ee	Bt	Mask	Token

マイクロインストラクションワードはデータワードと同じアクセプトにより支配される。

表 B. 2. 7 7 フィールドから成るマイクロインストラクションワード

B. 2. 3. 2 「動作モード」

50 【1314】

【表187】

Cmd	モード
0	Data Word
1	Data Token

表B. 2. 8 ビットアロケーション

B. 2. 3. 2. 1 「データワード」

本モードでは、入力の最上位8ビットが出力に送られる。底部8ビットはマスクフィールドに応じて、入力の底部8ビット、マイクロインストラクションワードのトークンフィールド、もしくはその両者の混合物のいずれかである。マスクは、

```
out data [16:8] = in data [16:8]
```

```
out data [7:0] = (Token [7:0]
& (ff<<mask)) in data [7:0]
```

のミックスにおける入力ビット数を表す。

【1315】マスクが0x8以上に設定された場合、出力データは入力データに等しい。本モードはnon-データトークンにおける出力ワードに用いられる。マスクを0に設定すると、out data [7:0] はマイクロインストラクションワードのトークンフィールドとなる。本モードは如何なるデータも含まないトークンヘ

```
If (first coefficient)
```

```
{
```

```
out data [16:2] = 0x1
```

```
out data [1:0] = Bt [1:0]
```

```
RL [16:0] = in data [16:0]
```

```
}
```

```
else
```

```
{
```

```
out data [16:0] = RL [16:0]
```

```
RL [16:0] = in data [16:0]
```

```
}
```

```
out extn = -Eb
```

の準備が整うことを確実にする。

【1317】B. 2. 3. 3 「注釈」

本発明によれば、指令ビットのほとんどは通常の方法でパーザステートマシンによって供給される。しかしながら、実際には2つのフィールドが他の回路により供給される。上述のBtフィールドはALUブロックの出力に直接接続される。この2つのビットフィールドはcもしくはcolor componentの現在のバリューを与える。こうして、データトークンヘッダが構築されると、最も低いオーダーの2ビットがALUカウンタから直接色成分を取り出す。第2に、ブロック・エンド記号idがデコードされる時はいつでも（あるいはブロック内の最後の係数がコード化されるので、1が仮定されるJPEGの場合）、Ebビットがハフマンデ

ッダを出力するために使用される。トークンヘッダがデータを含む場合、データビット数はマスクフィールドにより与えられる。

【1316】外部Extn (Ee) が設定されると、out extn=in extnとなり、そうでなければ、out extn=De. Btであり、Ebはdon't careである。

B. 2. 3. 2. 2 「データトークン」

本モードはデータトークンとフォーマットングするために使用され、信号、first coefficient に応じて2つの関数がある。リセットにおいて、first coefficient が設定される。最初のデータ係数がcmdを1に設定したマイクロインストラクションワードと共に到着すると、out data [16:2] は0x1に設定され、out data [1:0] はマイクロインストラクションワードのBtフィールドの値を取る。これはデータトークンのヘッダである。このワードがアクセプトされると、コマンドを伴う係数がレジスタRLにロードされ、first coefficient がEbの値を取る。次の係数が到着すると、out data [16:0] はRLに記憶された前の係数を取る。これはブロック・エンドに遭遇する時、Ebが設定され、first coefficient が設定され、次のデータトークン、つまり

ードにおいて認定される。

【1318】ハフマンデコードにおいてin extn 信号が引き出される。それはトークンに関して、通常の方法でトークンワードと共に拡張ビットが供給される時に意味を持つだけである。

【1319】B. 2. 4 「パーザステートマシン」

本発明のパーザステートマシンは実際非常に単純な回路片である。複雑なのはマイクロコードROMのプログラミングであり、それはセクション B. 2. 5において論じる。

【1320】本質的に、マシンは現在のアドレスを保持するレジスタから構成される。このアドレスはマイクロコードROMにおいてルックアップされ、マイクロコ

ードワードを作り出す。更に、アドレスは単純なインクリメントにおいて増分され、この増分されたアドレスは次のステートのために使用される2つの可能なアドレスの1つである。他方のアドレスはマイクロコードROM自体の中のフィールドである。このように、各指令は潜在的にジャンプ指令であり、プログラムにおいて明記されるロケーションにジャンプすることができる。ジャンプが行われなければ、制御はROM内の次のロケーションへと進む。

【1321】一連の16のコンディションコードビット 10
が提供される。これらのコンディションの1つが（マイクロコードROM内のフィールドにより）選択され、そ

れに加えて、（やはりマイクロコードROM内のビットにより）反転され得る。結果的に生じる信号が増分されたアドレスまたはマイクロコードROM内のジャンプアドレスのいずれかを選択する。コンディションの1つはFalseとして評価するため配線される。このコンディションが選択されると、ジャンプは発生しない。あるいはその代わりに、このコンディションが選択され、反転されて、ジャンプ、つまり無条件ジャンプが常に発生する。

【1322】
【表188】

ビット数	名称	説明
0	use [0]	マイクロプロセッサインターフェイスからユーザーがプログラムできるレジスタに接続される。それらは殆ど種族を使わずにテストできる「ユーザー限定」コンディションコードを可能にする。試験的な4ブロック×8ブロックのマクロブロック構造のための非標準「コード化ブロックパターン」処理を制御するために2つが限定される。
1	use [1]	
2	cbp eight	
3	cbp special	
4	he [0]	これらのビットはハフマンデコードのハフマン誤差レジスタに直接接続する。
5	he [1]	
6	he [2]	
7	Ext n	(トークン用の) 拡張ビット
8	Blkptn	ブロックパターンシフト
9	MBstart	マクロブロック・スタートにおいて
10	Picstart	ピクチャ・スタートにおいて
11	Restart	リスタート間隔のスタートにおいて
12	Chngdet	「スティッキー」変化検出ビット
13	Zero	ALUゼロ状態
14	Sign	ALUサイン状態
15	False	False (ハードワイヤード)

表B. 2. 9 コンディションコードビット

B. 2. 4. 1 「2線式インターフェース制御」

本発明による2線式インターフェース制御は本ブロックでは少し異例である。パーザーステートマシンとハフマンデコード間に2線式インターフェースがある。これはコマンドの進行を制御するために使用される。パーザーステートマシンはROMから次のコマンドを読むために前進する前に、所定のコマンドがアクセプトされるまで待つであろう。それに加えて、コンディションコードがALUからのワイヤを通して送り返される。

【1323】各コマンドはマイクロコードROM内にビットを持ち、それはフィードバックを待つべきことを明記できるようにする。これが発生した場合、その指令がハフマンデコードによりアクセプトされた後、ALUからのフィードバックワイヤが認定されるまで新しいコマンドは表示されない。このワイヤ、fb validは現在ALUにより供給されているコンディションコードが、それらがフィードバックを待つことを求めるコマンドと関連するデータを反映するという意味で有効であることを指示する。

【1324】本発明によれば、デコーディング（またはプロセッシング）の結果として、特別なデータ片にジャン

30 プするための次のステートを決定する条件付きジャンプコマンドを構築する際に、その特徴が利用される。2線式制御は特定のコマンドが所定のプロセッシングブロック（つまり、この場合はALU）に到達する時が不確かであることを意味するので、この便宜なくしては、パイプライン内のデータに依存するコンディションをテストすることは出来ないであろう。

【1325】ハフマンデコードに全ての指令が送られるわけではない。ある指令はデータパイプラインの必要なく実行できる。これらはジャンプ指令である傾向がある。マイクロコードROM内のビットは指令がハフマンデコードに表示されるか否かを選択する。そうでなければ、ハフマンデコードが指令をアクセプトする必要はなく、従って、パイプラインが失速していても、こうした状況で実行を続けることができる。

【1326】B. 2. 4. 2 「イベント処理」
パーザーステートマシンの中に2つのイベントビットが置かれる。1つはハフマンイベントと称され、他の1つはパーザーイベントと称される。

【1327】パーザーイベントは最も単純である。このイベントにより監視される「コンディション」はマイク

ロコードROM内のビットにすぎない。このように、指令はこのビットを設定することによりパーザイベントを生じさせることができる。典型的に、これを行う指令は適当な定数をrom controlレジスタに書き込み、割り込みサービスルーチンが割り込みの原因を判断することができるようにする。

【1328】パーザイベントをサービスした後（あるいは、イベントがマスクアウトされた場合直ちに）、制御は停止したポイントでリカバーする。そのイベントを起こさせた指令がジャンプ指令を持っている（その状態が本物であると評価する）場合、ジャンプは通常の方法で行われる。従って、ジャンプをコード化することによるサービス後、誤差ハンドラにジャンプすることができる。

【1329】ハフマンイベントはそれとは異なる。監視される状態は3つのハフマン誤差ビットのORである。現実には、この状態は非常に単純な方法でパーザイベントに対して処理される。しかしながら、ハフマンデコードからの追加ワイヤ、huffintrptは誤差が発生する時はいつでも認定される。これはマイクロコードプログラムにおいて制御を誤差ハンドラにジャンプさせる。

【1330】従って、ハフマン誤差が発生する時、シーケンスは割り込み発生とブロック・ストップを含む。サービシング後、制御は誤差ハンドラに伝送される。callメカニズムは無く、通常の割り込みとは異なり、誤差処理に続いて発生する誤差の前にマイクロコードの中のポイントに戻ることは不可能である。

【1331】ハフマン誤差が発生することなく、huffintrptを認定することができる。これはセクション B. 2. 2. 3において論じたように、noerror誤差の特別な場合に発生する。この場合、如何なる割り込みも（マイクロプロセッサインターフェースに）発生せず、制御は（マイクロコード内の）誤差ハンドラに送られる。ハフマン誤差レジスタはこの場合明らかであるので、マイクロコード誤差ハンドラは、こうした状況であり、それに応じて応答することを決定することができる。

【1332】B. 2. 4. 3 「特別なロケーション」マイクロコードROMには幾つかの特別なロケーションがある。ROM内の最初の4つのロケーションは主プログラムへのエントリポイントである。制御はリセット時にこれら4つのロケーションの1つに進む。ジャンプすべきロケーションはALUレジスタ、coding std. において選択されるコーディングスタンダードにより決定される。このロケーションが真のリセットによりそれ自体ゼロにリセットされるので、制御はロケーションゼロに進む。しかしながら、CED H TRACE RSTを用いてパーザステートマシンだけをリセッ

トすることができる。この場合、coding stdレジスタはリセットされず、制御は最初の4つのロケーションの適当な1つに進む。

【1333】第2の4つのロケーション（0x004～0x007）はハフマン割り込みが起こる時に使用される。典型的に、実際の誤差ハンドラがこれらのロケーションの各々に置かれる。ここでも、ロケーションの選択はコーディングスタンダードの結果として為される。

【1334】B. 2. 4. 4 「トレーシング」診断上の助けとして、トレースメカニズムが実装される。これはマイクロコードがシングルステップ化されるようにする。レジスタCED H TRACE内のビットCED H TRACE EVENTとCED H TRACE MASKがこれを制御する。その名前が示すように、それらは非常に良く似た方法で通常のイベントビットに対して作用する。しかしながら、幾つかの違い（特に、UPI割り込みが生じないこと）により、それらは他のイベントビットと一緒にグループ分けされることはない。

【1335】CED H TRACE MASKが1に設定される時にトレーシングメカニズムがオンにされる。各マイクロコード指令がROMから読まれた後、しかしそれがハフマンデコードに表示される前に、トレースイベントが発生する。この場合、CED H TRACE EVENTが1になる。如何なる割り込みも生じないので、それが登録されなければならない。全マイクロコードワードがレジスタCED H KEY DMXWORD 0～CED H KEY DMXWORD 9において利用できる。その指令は必要であれば、この時点で修正できる。CED H TRACE EVENTに1を書き込むと、その指令を実行させ、CED H TRACE EVENTをクリアする。この後すぐに、実施すべき次のマイクロコードワードがROMから読み出され、新しいトレースイベントが発生する。

【1336】B. 2. 5 「マイクロコード」マイクロコードはアセンブラhppを用いてプログラムされるが、それは非常に単純なツールで、抜取りの大部分がマクロプリプロセッサを使用して行われる。標準のCプリプロセッサcppがこの目的のために使用される。

【1337】コードは次のように指示される。

【1338】Ucode. uは主ファイルである。まず、これはトークンを定義するtokens. hを含む。次に、regfile. hはALUレジスタmapを定義する。fields. uはマイクロコードワード内の様々なフィールドを定義し、フィールド内の各々の可能なビットパターンのために定義された記号リストを提供する。次に、コードの中で使用されるラベルが定義される。このステップの後、基本的な指令を明確にする多数のcppmacrosを定義するために、inst

r uが含まれる。次に、errors. hがパーザイベントを定義するカンバーを定義する。次に、unword. uはマイクロコードワードを構築するためにフィールドが置かれるオーダーを定義する。

【1339】残りのucode. uはマイクロコードプログラム自体である。

【1340】B. 2. 5. 1 「指令」

本セクションでは、ucode. uにおいて定義される様々な指令を説明する。多くの場合、1つのテーマ（特にALU指令）に関して変形が少ないので、全ての指令 10 に関する説明をここで行うわけではない。

【1341】B. 2. 5. 1. 1 「ハフマン及びデータインデックス指令」

本発明では、ハフマンデコーダはH NOP指令を使用する。それはノーオペレーション指令である。ハフマンは如何なるデータもデコードされないという意味において、何もしない。この指令により作られるデータは常にゼロである。従って、関連指令はALUに送られる。

【1342】次の指令はトークングループ；H TOKSRCH、H TOKSKIP PAD、H TOKS 20 KIP JPAD、H TOKPASS及びH TOKREADである。これらは総て入力シフタからのトークンを読み出し、それらを残りのマシンに送る。H TOKREADは1つのトークンワードを読む。H TOKPASSはゼロextnビットまでの全トークンを読むために使用できる。関連コマンドはトークンの各ワードのために繰り返される。H TOKSRCHはトークンの前の全てのシリアルデータを捨て、1つのトークンワードを読む。H TOKSKIP PADはパディングビット（H. 261及びMPEG）をとばし、1つの 30 トークンワードを読む。H TOKSKIP JPADはJPEGパディングのために同じことをする。

【1343】H FLC (NB) はNBビットの固定長コードを読む。

【1344】H VLC (TBL) は (mnemonic、例えばH VLC (tcoeff) として送られる) 指示されたテーブルを用いて、vicを読む。

【1345】H FLC TE (NB) はH FLCと同様であるが、ignore errorsビットが設定される。

【1346】H TEST VLC (TBL) はH FLCと同様であるが、ハフマンインデックスが無修正のデータユニットインデックス部を通して送られるように、バイパスビットが設定される。

【1347】H FWD R及びH BWD RはALUレジスタr fwd r sizeとr bwd r sizeにより各々指示されるサイズのFLCを読む。

【1348】H DCJはJPEGスタイルのDC係数、ALUからのテーブルナンバを読む。

【1349】H TC/OE/FF及びH DCTC/OE/FFは変換係数を読む。HDCTC/OE/FFにおいて、第1のcoeffビットが設定され、non-intraブロックのためのものである一方、H TC/OE/FFはDCタームが読まれた後のintraブロックのためである。

【1350】H NOMINATE (TBL) は次のダウンロードのためのテーブルを指名する。

【1351】H DNL (NB) はNBビットを読み、それらを指名されたテーブルにダウンロードする。

【1352】B. 2. 5. 1. 2 「ALU指令」
実際には、詳細に説明するにはあまりに多くのALU指令がありすぎる。Mnemonics構築される基本的な方法について論じ、これにより指令が読めるようになるべきである。更に、これらは当業者の一人であれば容易に理解できるものでなければならない。

【1353】多くのALU指令は場所から場所へとデータを移動させることに関係し、従って、一般的な「ロード」指令が使用される。Mnemonicにおいて、ALDxyは、yの内容がxにロードされる、つまりデスティネーションが先に記録され、ソースが次にくることが解る。

【1354】

【表189】

文字	意味
A	Aレジスタ
R	Runレジスタ
I	データ入力
O	データ出力
F	ALUレジスタファイル
C	定数
Z	ゼロの定数

表B. 2. 10 可能なソース及びデータのデスティネーションを表示するための文字

例として、LDAIはAレジスタにALUのデータ入力ポートからのデータをロードする。ALUレジスタファイルが指定されると、ニマニックはLDAF (RA) がレジスタファイル内のロケーションRAの内容をAにロードするようにアドレスを取るであろう。

【1355】ALUはソースからデスティネーションに動かされるにつれて、データを修正する能力を有する。この場合、算術はソースデータの一部として指示される。従って、Mnemonic LDA AADD (RA) は、Aレジスタの現在の内容プラス、レジスタファイル内の指示されたロケーションの内容をAにロードする。別の例はLDA ISGXRであり、それは入力データを取り、サインはRUNレジスタの中で指示されるビットから伸び、Aレジスタにその結果を記憶する。

【1356】多くの場合、同じ結果のために1つ以上のデスティネーションが指定される。ここでも、例として、LDF LDA ASUBC (RA) では、Aマイナす定数の結果をAレジスタ及びレジスタファイルの両方にロードする。

【1357】他のニマニックスは特別な動作のために存在する。例えば、CLRAはAレジスタをクリアするために使用され、RMBCはマクロブロックカウンタをリセットするために使用される。これらはかなり自明であり、instr. uの中のコメントにおいて説明される。

【1358】1つの例外は、オペレーションの結果が通常の動作に加えてトークンフォーマットに出力されることを指示するために、接尾辞Oを使用することである。このように、LDF IO (RA) は入力データを記憶し、更にそれをトークンフォーマットに送る。あるいはその代わりに、所望であれば、これはLDF LDO I (RA) であってもよいであろう。

B. 2. 5. 1. 3 「トークンフォーマット指令」

これはT NOP「ノーオペレーション」指令である。

これは実際には、ノーオペレーション指令を構築することは不可能なので、誤称である。しかしながら、これはALUがトークンフォーマットに出力されないもので、その指令が重要ではない時に使用される。

【1359】T TOKはトークンワードを出力する。

【1360】T DATは(ハフマンステートマシン指令と共にのみ使用される) データトークンワードを出力する。

【1361】T GENT8は一定フィールドの8ビットに基づいてトークンワードを生み出す。

【1362】T GENT8EはT GENT8と同様であるが、拡張ビットは1である。T OPD (NB) は一定フィールドから来るビットの残部と共に、出力の底部NBビットからのデータのNBビット。

【1363】T OPDE (NB) はT OPDと同様であるが、拡張ビットが高い。

【1364】T OPD8はT OPD (8) 用のショートハンドである。

【1365】T OPD8EはT OPDE (8) 用のショートハンドである。

【1366】B. 2. 5. 1. 4 「パーザステートマシン指令」この指令、D NOPはノーオペレーション、つまり、アドレスは普通に増加し、パーザステートマシンは何も特別なことをしない。指令の残部はデータパイプラインに送られる。待機は発生しない。

40 【1367】D WAITはD NOPと同様であるが、フィードバックが発生するのを待つ。

【1368】単純なジャンプグループ。D JMP (ADDR) 及びD JNX (ADDR) のようなニマニックスは、条件が満たされればジャンプする。指令はハフマンデコードに出力されない。

【1369】外部ジャンプグループ。D XJMP (ADDR) 及びD XJNX (ADDR) のようなニマニックス。これらは上記の単純な相対物と同様であるが、その指令はハフマンデコードに出力される。

50 【1370】ジャンプ及び待機グループ。D WJNZ

(ADDR)等のニマニックス。これらの指令はハフマンデコードに出力され、パーザは状態を評価する前に、ALUからのフィードバックを待つ。

【1371】以下のニマニックスは状態自体のために使

用される。

【1372】

【表190】

ニマニックス		意 味
JMP	.	無条件ジャンプ
JXT	JNX	extn=1 (extn=0) であればジャンプ
JHE0	JNHE0	ハフマン誤差ビット0セット (クリア) であればジャンプ
JHE1	JNHE1	ハフマン誤差ビット1セット (クリア) であればジャンプ
JHE2	JNHE2	ハフマン誤差ビット0セット (クリア) であればジャンプ
JPTN	.	パターンシフトLSBが設定されればジャンプ
JPICST	JNPICST	ピクチャ・スタートであれば (ピクチャ・スタートでなければ) ジャンプ
JRSTST	JNRSTST	リスタート間隔のスタートであれば (スタートでなければ) ジャンプ
.	JNCPBS	特別なCPBコーディングでなければジャンプ
.	JNCPB8	8ブロックではない (つまり、4ブロックの) マクロブロックであればジャンプ
JMI	JPL	マイナスであればジャンプ (プラスであればジャンプ)
JZE	JNZ	ゼロであればジャンプ (ゼロでなければジャンプ)
JCHNG	JNCHNG	変化検出ビットセット (クリア) であればジャンプ
JMBST	JNMBST	マクロブロック・スタートであれば (スタートでなければ) ジャンプ

表B. 2. 11 状態のために使用されるニマニックス

D EVENTはイベントの発生を引き起こす。

【1373】D DELTはデフォルト指令の構築のためである。これはイベントを引き起こし、ラベルdf1tを持つロケーションにジャンプする。この指令はROMを満たすために使用され、未使用のロケーションがトラップされるので、実行されてはならない。

【1374】D ERRORはイベントを引き起こし、誤差からのリカバリを企てると仮定されるラベルsrch dispatchにジャンプする。

セクションB. 3 「ハフマンデコードALU」

B. 3. 1 「序文」

本発明によれば、ハフマンデコードALUサブブロックはハフマンデコードブロック用の全般的な算術的及び論理的機能性を提供する。それは加算及び減算操作、様々なタイプのサイン拡張操作、及び入力データのrunsign-levelトリプルズへのフォーマットを行う能力を有する。更に、それはその正確な操作及び構成が、入力データと同様に、つまり2線式インターフェース制御下に、ALUに到着するマイクロインストラクションワードにより明記される柔軟な構造を有する。

【1375】36ビットの指令及び12ビットデータ入

力ポートに加えて、ALUは6ビットランポート及び (実際にはトークンバスの上にある) 8ビット一定ポートを持つ。これらの全ては、マイクロインストラクションワードを除き、ALUデータバスを通して各々の幅のバスを駆動する。拡張ビットを表し、17ビットラン・サイン・レベルと共に出力される (out data) マイクロインストラクションワード内に1ビットがある。ALUデータバスの各端に2線式インターフェースと、それら自身の有効な信号、cc validと共に出力される一連のコンディションコードがある。ALUを介して他のハフマンデコードサブブロック及びマイクロプロセッサインターフェースにアクセス可能なレジスタファイルがある。

【1376】B. 3. 2. 2 「基本構造」

ハフマンALUの基本構造は図135に示す通りである。それは以下の構成要素を含む：

入力ブロック

出力ブロック

コンディションコードブロック

ソースマルチプレクシングを備えたAレジスタ

ソースマルチプレクシングを備えたRunレジスタ (6

ビット)

415

ソースマルチプレクシングを備えたアダー/サブトラクター

ソースマルチプレクシングを備えたサイン拡張ロジックレジスタファイル

(出力ブロックを除き) これらのブロックの各々はその出力をデータバスを通して走るバス上に押しやり、これらのバスは次にブロックソース用のマルチプレクシングへの入力として使用される。例えば、アダーはAレジスタに対する可能な入力の1つである、それ自身のデータバスバスを有する。同様に、Aレジスタはアダーに対する可能な入力の1つを形成するそれ自身のバスを有する。この点で、マイクロインストラクションワードに関するセクション7において明記したように、全ての可能性のサブセットだけが存在する。

【1377】単一サイクルにおいて、アッドベースの指令もしくはサイン拡張ベース指令のいずれかを実行することが可能である。更に、それらのオペレーションが厳密に平行である場合に限り、それらの両方を単一サイクルにおいて実行してもさしつかえない。換言すれば、加算してサイン拡張、あるいはサイン拡張して加算のシーケンスは許されない。レジスタファイルは単一サイクルにおいて読み取られるか書き込まれるが、その両方ではない。

【1378】出力データは3つのフィールドを持つ：

- ・ラン - 6ビット
- ・サイン - 1ビット
- ・レベル - 10ビット

データがALUを通してまっすぐ送られる場合、入力データレジスタの最も重要性の少ない11ビットがサイン及びレベルフィールドにラッチされる。

【1379】制限されたALUのマルチサイクルオペレーションをプログラムすることができる。この点で、必要なサイクル数はレジスタファイルロケーションの内容により決められ、そのアドレスはマイクロインストラクションにおいて明記され、反復カウンタが1にまで減少する間に、同じオペレーションが繰り返し実行される。この設備は、典型的に、左方向のシフトを行うために使用され、アダーを用いてAレジスタをそれ自体に加算し、その結果をAレジスタに戻して記憶する。

【1380】B. 3. 3 「アダー/サブトラクターサブブロック」

これは12ビット幅のアダーであり、そのinput 2への任意反転と、carry-inビットの任意セッティングを持つ。出力は12ビットの合計であり、carry-outは使用されない。オペレーションには7つのモードがある：

・ADD：セットのキャリーをゼロに加算する：

input 1 + input 2

・ADC：セットのキャリーを1に加算する：

input 1 + input 2 + 1

416

・SBC：input 2を反転し、セットのキャリーをゼロに：

input 1 - input 2 - 1

・SUB：input 2を反転し、セットのキャリーを1に：

input 1 - input 2

・TCI：input 2 < 0であれば、SUBを使用し、そうでなければADDを使用する。これは2の補数値からマグニチュードバリューを得るために、ゼロに設定されるinput 1と共に使用される。

【1381】DCD (DC差)：input 2 < 0であれば、ADCを行い、そうでなければADDを行う。

【1382】VRA (ベクトル残差アッド)：input 1 < 0であれば、ADCを行い、そうでなければSBCを行う。

【1383】B. 3. 4 「サイン拡張サブブロック」これは様々なモードでサインがサイズ入力からの入力データを拡張する12ビットユニットである。サイズは0～11までの4ビットバリューである(0は最も重要でないビットに関し、11は最も重要なビットに関する)。出力は12ビット修正データバリューであり、「サイン」ビットである。

【1384】SGXMODE=NORMALでは、size-thビット以上の全てのビットは、size-thビットの値を取る。下記のもの全ては変化せずそのままである。サインはsize-thビットの値を取る。例えば：

data = 1010 1010 1010

size = 2

output = 0000 0000 0010, sign=0

SGXMODE=INVERSEでは、size-thビット以上の全てのビットは、size-thビットの逆数を取る一方、下記のもの全ては変化せずそのままである。サインはsize-thビットの逆数を取る。例えば：

data = 1010 1010 1010

size = 0

output = 1111 1111 1111, sign=1

SGXMODE=DIFMAGでは、size-thビットがゼロであれば、size-thビット以下の全てのビットは反転される一方、上記の全てのは変化せずそのままである。size-thビットが1であれば、全てのビットは変化せずに残る。両方の場合に、サインはsize-thビットの逆数を取る。これはAC差バリューのマグニチュードを得るために使用される。

例えば：

data = 0000 1010 1010

size = 2

417
 output = 0000 1010 1101, s
 ign=1
 data = 0000 1010 1010
 size = 1

output = 0000 1010 1101, s
 ign=0

SGXMODE=DIFCOMPでは、size-th
 ビット以上の（ではあるがsize-thビットは含ま
 ない）全てのビットはsize-thビットの逆数を取
 り、一方それ以下の全てのビットは変化せずに残る。サ
 インはsize-thビットの逆数を取る。これはDC
 差バリュウのために2つの補数値を取るために使用され
 る。例えば：

data = 1010 1010 1010
 size = 0
 output = 1111 1111 1110, s
 ign=1

B. 3. 5 「コンディションコード」

ハフマンブロックが使用するコンディションコードの2
 つのバイト（16ビット）があり、その内特定のビット
 がALU/レジスタファイルによって作られる。これら
 はサインコンディションコード、ゼロコンディションコ
 ード、拡張コンディションコード、及び変化検出ビット
 である。これらのコードの最後の2つは、パーザーによ
 って他のものと同じようには使用されないで、実際に
 はコンディションコードではない。

【1385】サイン、ゼロ、及び拡張コンディションコ
 ードはパーザーが更新するように指令を出す時に更新さ
 れ、これら各々の指令のために、コンディションコード
 有効信号が一度だけ高いパルスで送られる。

【1386】サインコンディションコードは単にラッチ
 されるサイン拡張サイン出力であり、一方、ゼロコン
 ディションコードはAレジスタへの入力为零である場合
 に、1に設定される。拡張コンディションコードはOU
 T SRCとは無関係にラッチされる入力拡張ビットであ
 る。

【1387】コンディションコードは特定のコンディシ
 ョンタイプを評価するために使用され得る：

- ・結果は定数に等しい — サブトラクトを使用し、ゼ
 ロコンディション
- ・結果はレジスタ値に等しい — サブトラクトを使用
 し、ゼロコンディション
- ・レジスタは定数に等しい — サブトラクトを使用
 し、ゼロコンディション
- ・レジスタビットセット — サイン拡張を使用し、サ
 インコンディション
- ・結果ビットセット — サイン拡張を使用し、サイン
 コンディション

サイン拡張及びサインコンディションコードの組合せを
 使用する場合、従来の論理ANDの場合のように多重ビ

ットを評価するというより、1つの特定のビットを評価
 することだけが可能であることを注目。

【1388】本発明において、変化検出ビットはゼロコ
 ンディションコードとして同じロジックを使用して作ら
 れるが、関連する有効な信号を持ってはいない。マイク
 ロインストラクション内のビットは、レジスタファイル
 に最近書き込まれたバリュウが既に存在しているものと
 異なる（2つのクロックサイクルが必要であり、まずR
 EG-MODEをREADに設定し、次にREGMOD
 EをWRITEに設定することを意味する）場合に、変
 化検出ビットを更新すべきであることを指示する。次
 に、変更されたバリュウが検出された場合、マイクロプ
 ロセッサの割り込みが開始される。変化検出ビットは通
 常の方法でChange Detectを起動させることによりリセ
 ットされるが、REGMODEはREAD
 に設定される。

【1389】（レジスタファイルの一部を形成する、下
 記参照）ハードワイヤードマクロブロックカウンタ構造
 も、次のようなコンディションコードを作り出す：Mb
 Start、Pattern Code、Resta
 rt及びPic Start。

【1390】B. 3. 6 「レジスタファイル」

レジスタファイル用のアドレスマップを下記に示す。そ
 れはALUデータバス及びUPIの両方に共通である7
 ビットアドレススペースを使用する。多くのロケーショ
 ンはALUによってアクセスすることができず、これら
 は典型的にハードワイヤードマクロブロック構造の中
 のカウンタであり、ALU自体の中のレジスタである。後
 者のレジスタは専用のアクセスを持っているが、UPI
 用のアドレスマップの一部を形成する。（表において0
 によってオーバーサイズが表示される）あるマルチバイ
 トのロケーションは、1つのALUアドレスと、多重U
 PIアドレスを持つ。同様に、成分カウント、CC（表
 ではIで指示される）によりインデックスされるレジス
 タグループは、ALUにより1つのロケーションとして
 処理される。これは初期設定及びリセットtingのた
 め、またブロックレベルのオペレーションのためにマイ
 クロプログラミングを容易にする。

【1391】専用ALUレジスタ（UPIリードオンリ
 ー）を除き、全てのロケーションはリード/ライトであ
 り、全てのカウンタは指令ワード内のビットによってゼ
 ロにリセットされる。パターンコードレジスタは右方向
 シフトの能力を持ち、その最も重要でないビットがPa
 ttern Codeコンディションビットを形成す
 る。ハードワイヤードマクロブロック構造の全てのレジ
 スタは表中Mで表示され、更にカウンタ（n-ビット）
 であるものはCnという注釈が付けられる。

【1392】本発明では、特定のロケーションはハフマ
 ンサブシステムコーディングスタンダードの他の部分、
 2つのr-sizeロケーション、及びハフマンデコー

ダに対するac_huffテーブル及びdc_huffテーブルの各々のための1つのロケーション(1ビットワード)にハードワイヤードされる内容を持つ。

【1393】太字のアドレスはロケーションがALU及びUPIの両方でアクセス可能であることを示し、そうでなければ、それらはUPIアクセスだけを持つ。ALUによりCCを通して指示されないレジスタグループは、指令ワードの中で明記される1つのALUアドレスを持つことができ、CCはアクセスすべきグループ内の物理的ロケーションを選択するデータトークンであろう。ALUアドレスは、従来は最初のアドレスを使用すべきであったが、グループ内のどのレジスタのものであってもよい。これは、実際にはどちらのアドレスであっても充分であるが、ペアのアドレスのうち低い方のアドレス

を使用して、アクセスされるべきマルチバイトのロケーションの場合にもあてはまる。ロケーション2E及び2Fはトップレベルのアドレスマップ(Tで表示される)において、つまりキーホールレジスタを通してだけではなく、アクセス可能であることに注目。これら2つのロケーションもゼロにリセットされる。

【1394】レジスタファイルはアクセススピードを改良するため、4つの「バンク」に物理的に区画されるが、これは如何なる方法でもアドレッシングに影響を及ぼさない。主テーブルはMPEG用のアドレスマップを示し、2つの繰り返されるセクションがJPEGとH. 261用のバリエーションを各々提供する。

【1395】

【表191】

アドレス	ロケーション
00	A register 1
01	A register 0
02	run
10	horiz pels 1
11	horiz pels 0
12	vert pels 1
13	vert pels 0
14	buff size 1
15	buff size 0
16	pel asp. ratio
17	bit rate 2
18	bit rate 1
19	bit rate 0
1A	pic rate
1B	constrained
1C	picture type
1D	H261 picture type
1E	broken closed
1F	pred mode

表B. 3. 1 ハフマンレジスタファイルアドレスマップ (1/6)

【1396】

【表192】

アドレス	ロケーション
20	vbv delay 1
21	vbv delay 0
22	full pel fwd
23	full pel bwd
24	horiz mb copy
25	pic number
26	max h
27	max v
28	.
29	.
2A	.
2B	.
2C	first group
2D	in picture
T. R	2E rom control
T. R	2F rom revision

表B. 3. 1 ハフマンレジスタファイルアドレスマップ (2/6)

【1397】

【表193】

	アドレス	ロケーション	
I. H	30	dc huff 0	
I	31	dc huff 1	
I	32	dc huff 2	
I	33	dc huff 3	
I. H	34	ac huff 0	
I	35	ac huff 1	
I	36	ac huff 2	
I	37	ac huff 3	
I	38	tq 0	
I	39	tq 1	
I	3A	tq 2	
I	3B	tq 3	
I	3C	c0	
I	3D	c1	
I	3E	c2	
I	3F	c3	

表B. 3. 1 ハフマンレジスタファイルアドレスマップ (3/6)

[1398]

[表194]

	アドレス	ロケーション	
I. O	40	dc pred 01	
I. O	41	dc pred 00	
I. O	42	dc pred 11	
I. O	43	dc pred 10	
I. O	44	dc pred 21	
I. O	45	dc pred 20	
I. O	46	dc pred 31	
I. O	47	dc pred 30	
O	50	prev mh f 1	
O	51	prev mh f 0	
O	52	prev mv f 1	
O	53	prev mv f 0	
O	54	prev mh b 1	
O	55	prev mh b 0	
O	56	prev mv b 1	
O	57	prev mv b 0	

表B. 3. 1 ハフマンレジスタファイルアドレスマップ (4/6)

[1399]

[表195]

	アドレス	ロケーション	
M	60	mb horiz cnt1	C13
M	61	mb horiz cnt0	
M	62	mb vert cnt1	C13
M	63	mb vert cnt0	
M	64	horiz mb 1	
M	65	horiz mb 0	
M	66	vert mb 1	
M	67	vert mb 0	
M	68	restart count1	C16
M	69	restart count0	
M	6A	restart gap1	
M	6B	restart gap0	
M	6C	horiz blk count	C2
M	6D	vert blk count	C2
H. M	6E	compid	C2
M	6F	max compid	

表B. 3. 1 ハフマンレジスタファイルアドレスマップ (5/6)

[1400]

[表196]

	アドレス	ロケーション	
H. R	70	coding std	
M. H	71	pattern code	SR8
H	72	fwd size	
H	73	bwd size	
M. I	78	h0	
M. I	79	h1	
M. I	7A	h2	
M. I	7B	h3	
M. I	7C	v0	
M. I	7D	v1	
M. I	7E	v2	
M. I	7F	v3	

表B. 3. 1 ハフマンレグスタファイルアドレスマップ (6/6)

【1401】

【表197】

JPEG バリエーション:

	アドレス	ロケーション	
	10	horiz pels 1	
	11	horiz pels 0	
	12	vert pels 1	
	13	vert pels 0	
	14	buff size 1	
	15	buff size 0	
	16	pel asp. ratio	
	17	bit rate 2	
	18	bit rate 1	
	19	bit rate 0	
	1A	pic rate	
	1B	constrained	
	1C	picture type	
	1D	H261 picture type	
	1E	broken closed	
	1F	pred mode	

表B. 3. 2 JPEGバリエーション (1/2)

【1402】

【表198】

JPEG バリエーション:

	アドレス	ロケーション	
	20	vbv delay 1	
	21	vbv delay 0	
	22	pending frame ch	
	23	restart index	
	24	horiz mb copy	
	25	pic number	
	26	max b	
	27	max v	
	28	.	
	29	.	
	2A	.	
	2B	.	
	2C	first scan	
	2D	in picture	
	2E	rom control	
	2F	rom revision	

表B. 3. 2 JPEGバリエーション (2/2)

【1403】

【表199】

H. 261 バリエーション

アドレス	ロケーション
10	horiz pels 1
11	horiz pels 0
12	vert pels 1
13	vert pels 0
14	buff size 1
15	buff size 0
16	pel asp. ratio
17	bit rate 2
18	bit rate 1
19	bit rate 0
1A	pic rate
1B	constrained
1C	picture type
1D	H261 picture type
1E	broken closed
1F	pred mode

表B. 3. 3 H. 261バリエーション (1/2)

【1404】

【表200】

H. 261 バリエーション:

アドレス	ロケーション
20	vbv delay 1
21	vbv delay 0
22	full pel fwd
23	full pel bwd
24	horiz mb copy
25	pic number
26	max h
27	max v
28	.
29	.
2A	.
2B	.
2C	first group
2D	in picture
2E	rom control
2F	rom revision

表B. 3. 3 H. 261バリエーション (2/2)

B. 3. 7 「マイクロインストラクションワード」
 本発明によれば、ALUマイクロインストラクションワードは多くのフィールドに分けられ、その各々が上記構造の異なる局面を制御する。指令ワードの中で使用される全ビット数は36であり、(拡張ビット入力のためにはプラス1) 及びフィールドを横切る最小限度のエンコーディングが採用され、ハードウェア構成の最大限度の

柔軟性が維持される。指令ワードは下記に詳述するように区画される。デフォルトフィールドバリュー、つまり、ALUもしくはレジスタファイルのステートを変更しないものがイタリック体で示されている。

【1405】

【表201】

フィールド	バリュー	説明	ビット
OUTSRC (レジスタ出力)	RSA6	ラン、サイン、6ビットとレ アのレジスタ	0000
	ZZA	ゼロ、ゼロ、Aレジスタ	0001
	ZZA8	ゼロ、ゼロ、Aレジスタは8 ビットである	0010
	ZZADDU4	ゼロ、ゼロ、アダーo/p ms4ビット	0011
	ZINPUT	ゼロ、入力データ	0100
	RSSGX	ラン、サイン、サイン拡張 o/p	0111
	RSADD	ラン、サイン、アダーo/p	1000
	RZADD	ラン、ゼロ、アダーo/p	1001
	RIZADD	入力ラン、ゼロ、アダー出力	
	ZSADD	ゼロ、サイン、アダーo/p	1010
	ZZADD	ゼロ、ゼロ、アダーo/p	1011
	NONE	有効な出力がない - out validがゼロに 設定される	11XX
REGADDR	00.7F	ALUアクセス用のレジスタ ファイルアドレス	7ビット
REGSRC	ADD	レジスタファイルi/p上に アダーo/pを動かす	0
	SGX	レジスタファイルi/p上に サイン拡張o/pを動かす	1

表B. 3. 4 ハフマンALUマイクロインストラクション
フィールド (1/5)

[1406]

[表202]

フィールド	バリュー	説明	ビット
REGMODE	READ	レジスタファイルから読み出す	0
	WRITE	レジスタファイルに書き込む	1
CNGDET (change detect)	TEST	REGMODEがWRITEであれば、change detectを更新する	0
	HOLD	change detect ビットを更新しない	1
	CLEAR	REGMODEがREADであれば、change detectをリセットする	0
RUNSRC (ランソース)	RUNIN	ラン i / p をランレジスタ i / p 上に動かす	0
	ADD	アダー o / p をランレジスタ i / p 上に動かす	1
RUNMODE	LOAD	ランレジスタを更新する	0
	HOLD	ランレジスタを更新しない	1
ASRC (Aレジスタソース)	ADD	アダー o / p をAレジスタ i / p 上に動かす	00
	INPUT	入力データをAレジスタ i / p 上に動かす	01
	SGX	サイン拡張 o / p をAレジスタ i / p 上に動かす	10
	REG	レジスタファイル o / p をAレジスタ i / p 上に動かす	11

表B. 3. 4 ハフマンALUマイクロインストラクション
フィールド (2/5)

[1407]

[表203]

フィールド	バリュー	説明	ビット
AMODE	LOAD	Aレジスタを更新する	0
	HOLD	Aレジスタを更新しない	1
SGXMODE (サイン拡張 モード、セク ション4参照)	NORMAL	signでサイン拡張	00
	INVERSE	-signでサイン拡張	01
	DIFMAG	サインビットがゼロであれば 低い方のビットを逆転する	10
	DIFCOMP	次のビットアップから -signでサイン拡張	11
SIZESRC (サイン拡張 サイズ入力)	CONST	サイン拡張サイズi/p上に 定数i/pを動かす	00
	A	Aレジスタをサイン拡張 サイズi/p上に動かす	01
	REG	レジスタファイルo/pを サイン拡張サイズi/p上に 動かす	10
	RUN	ランレジスタをサイン拡張 サイズi/p上に動かす	11
SGXSRC (サイン拡張 サイズ入力)	INPUT	入力データをサイン拡張 サイズi/p上に動かす	0
	A	Aレジスタをサイン拡張 サイズi/p上に動かす	1

表B. 3. 4 ハフマンALUマイクロインストラクション
フィールド (3/5)

【1408】

【表204】

フィールド	バリュー	説明	ビット
ADDMODE (アダーモード セクション3参 照)	ADD	$input1 + input2$	000
	ADC	$input1 + input2 + 1$	001
	SBC	$input1 - input2 - 1$	010
	SUB	$input1 - input2$	011
	TCI	$input2 < 0$ であれば SUB、そうでなければ、 ADD-2の補数	100
	DCD	$input2 < 0$ であれば ADC、そうでなければ、 ADD-DC差	101
	VRA	$input1 < 0$ であれば ADC、そうでなければ、 SBC-vec resid add	110
ADDSRC1 (アダー入/ 出力1、逆数 なし)	A	Aレジスタをアダー $input1$ に動かす	00
	REG	レジスタファイルo/pを アダーi/plに動かす	01
	INPUT	入力データをアダー $input1$ に動かす	10
	ZERO	ゼロをアダー $input1$ に 動かす	11

表B. 3. 4 ハフマンALUマイクロインストラクション
フィールド (4/5)

【1409】

30 【表205】

フィールド	バリュー	説 明	ビット
ADDSRC2 (逆数入力用 ソース)	CONST	定数 i/p をアダー input 2 に動かす	00
	A	A レジスタ をアダー input 2 に動かす	01
	INPUT	入力データをアダー input 2 に動かす	10
	REG	レジスタファイル o/p を アダー input 2 に動かす	11
CNDC- MODE (コンディショ ンコード)	TEST	コンディションコードを更新 する	0
	HOLD	コンディションコードを更新 しない	1
CNTMODE (mb 構造カウ ントモード)	NOCOUNT	カウンタを増分しない	X00
	BCINCR	ブロックカウンタとリップル を増分する	001
	CCINCR	成分カウンタを incr に押 しやる	010
	RESET	mb 構造において全ての カウンタをリセットする	011
	DISABLE	全てのカウンタを不能化する	1XX
INST- MODE	MULTI	現在の instr を複数回繰 り返す	0
	SINGLE	シングルサイクル指令のみ	1

表B. 3. 4 ハフマンALUマイクロインストラクション
フィールド (5/5)

セクションB. 4 「バッファマネージャ」

B. 4. 1 「序文」

本文書は本発明によるバッファマネージャの目的、動作
及び実装を説明する (bman)。

【1410】 B. 4. 2 「展望」

バッファマネージャはDRAMインターフェース用に4
つのアドレスを提供する。これらのアドレスはDRAM
内のページアドレスである。DRAMインターフェース
はDRAM内に2つのFIFO、コード化データバッフ
ァとトークンデータバッファを維持する。従って、4つ
のアドレスのために、各バッファのためのリードアドレ
スとライトアドレスがある。

【1411】 B. 4. 3 「インターフェース」

バッファマネージャはDRAMインターフェースとマイ
クロプロセッサに対してのみ接続される。マイクロプロ
セッサはB. 4. 4において示される「初期設定レジス
タ」を設定するためだけに使用する必要がある。DRA
Mインターフェースを備えたインターフェースは、各ア
ドレスのためにREQUEST/ACKNOWLEDGEプロトコールにより制御される4つの18ビットアド
レスである。(バッファマネージャはデータバスの中に

30 はないので、バッファマネージャは2線式インターフェ
ースが欠けている。)

更に、バッファマネージャはDRAMインターフェース
クロック発生器をオフにし、DRAMインターフェース
スキャンチェインをオンにする。

【1412】 B. 4. 4 「アドレス計算」

各バッファ用のリードアドレス及びライトアドレスは9
個の18ビットレジスタから作られる:-

初期設定レジスタ (マイクロプロセッサからのRW)

・BASECB - コード化データバッファのベース
アドレス

・LENGTHCB - (コード化データバッファの
ページにおける) 最大サイズ

・BASETB - トークンデータバッファのベース
アドレス

LENGTHTB - トークンデータバッファの (ペ
ージにおける) 最大サイズ

・LIMIT - DRAMの (ページにおける) サイ
ズ

ダイナミックレジスタ (マイクロプロセッサからのR
0)

・READCB - BASECBに関するコード化データバッファリードポインタ

・NUMBERCB - READCBに関するコード化データバッファライトポインタ

・READTB - BASETBに関するトークンデータバッファリードポインタ

・NUMBERTB - READTBに関するトークンデータバッファライトポインタ

アドレスを計算するには:-

$readaddr = (BASE + READ) \bmod L$ 10
LIMIT

$writeaddr = ((READ + NUMBER) \bmod LENGTH) + BASE) \bmod LIM$
IT

バッファがDRAMのまわりを包むことがあるので、mod LIMITという用語が用いられる。

【1413】B. 4. 5 「ブロックの説明」

本発明では、バッファマネージャはスノーバがDRAMインターフェース接合部を監視するリングの中で接続される3つのトップレベルのモジュールで構成される。モ 20
ジュールはbmprtize (プライオリティ)、bminstr (指令) であり、bmrecalc (再計算) はそのオーダーのリングの中に配置され、omsnop (スノーバ) はアドレス出力の上に配置される。

【1414】モジュール、BmprtizeはREQ/ACKプロトコル、バッファ用のFULL/EMPTYフラグを処理し、各アドレスのステート、つまり、is it a valid address? を維持する。この情報から、それは (もしある場合) どのアドレスを再計算すべきであるかをbminstrに命じる。 30

更に、それはFULL/EMPTYフラグを示すBUF CSR (status) マイクロプロセッサレジスタを操作し、またマイクロプロセッサライトアクセスをバッファマネージャレジスタに対して制御する、buf access マイクロプロセッサレジスタを操作する。

【1415】モジュール、Bminstrは、bmprtizeによりアドレスを計算するように命じられると、6つの指令 (2サイクル毎に1つ) を出して、アドレスを計算するためにbmrecalcを制御する。

【1416】モジュール、Bmrecalcはbminstrの指令の下、アドレスを再計算する。それは2サイクル毎に指令を動かし、初期設定レジスタ及びダイナミックレジスタの全て及び加算、減算及びモジュラスができる単純なALUを含む。それはそれが検出するFULL/EMPTYステートのSbmprtizeを知ら 40

せ、それが完了すると、アドレスを計算する。

【1417】B. 4. 6 「ブロック実装」

B. 4. 6. 1 「Bmprtize」

リセットにおいて、buf access マイクロプロセッサレジスタは1に設定され、初期設定レジスタの設定を可能にする。buf access が1を読み戻している間は、如何なるアドレス計算も開始されない。なぜなら、有効な初期設定レジスタがなければ、それらの計算は無意味だからである。

【1418】一度buf access が完全認定される (それにゼロを書き込む) と、その目的は全ての4つのアドレスを有効にすることであるので、bmprtize は (それらを再計算して) 全てのアドレスを有効にするように務める。この段階で、バッファマネージャは「スターティングアップ」であり (つまり、全てのアドレスがまだ計算されていない) 、こうして如何なるリクエストも認定されない。全てのアドレスが一度有効になると、スタートアップが終了し、全てのリクエストが認定される。この時点から、アドレスが無効になる (一度使用され認識されているので) と、それは再計算されることになる。

【1419】アドレス間に優先順位を付けることは必要ではない。なぜなら、バッファマネージャが12サイクル毎にアドレスを再計算することができる一方、DRAMインターフェースは最高の場合で、各17サイクル毎にアドレスを使用することができるからである。従って、スタートアップ後、一度に1つのアドレスだけが無効となる。従って、bmprtize は現在計算されていない無効アドレスを再計算するであろう。

【1420】発明では、スタートアップはbuf access が認定される度に再入力されるので、マイクロプロセッサアクセス中は如何なるアドレスもDRAMインターフェースに供給されない。

【1421】B. 4. 6. 2 「Bminstr」

モジュール、BminstrはMOD12サイクルカウンタ (アドレスを作るためにそれが取るサイクル数) を含む。偶数サイクルが指令を開始するのに対して、奇数サイクルが指令を終了することに注目。それがリードもしくはライト計算であるか否かと共に、上位3ビットは次のように、bmrecalcのための指令にデコードされる:

リードアドレスのために:

【1422】

【表206】

サイクル	オペレーション	BusA	BusB	結果	結果のサインの意味
0-1	ADD	READ	BASE		
2-3	MOD	Accum	LIMIT	Ad- dress	
4-5	ADD	READ	" 1"		
6-7	MOD	Accum	LENGTH	READ	
8-9	SUB	NUMBER	" 1"	NUMBER	
10-11	MOD	" 0"	Accum		SET EMPTY (No. * >=0)

No. *: NUMBER

表B. 4. 1 リードアドレス計算

ライトアドレスのために:
【1423】

【表207】

20

サイクル	オペレーション	BusA	BusB	結果	結果のサインの意味
0-1	ADD	NUMBER	READ		
2-3	MOD	Accum	LIMIT		
4-5	ADD	Accum	BASE		
6-7	MOD	Accum	LIMIT	Ad- dress	
8-9	ADD	NUMBER	" 1"	NUMBER	
10-11	MOD	Accum	LENGTH		SET FULL (No. * >= LNG*)

No. *: NUMBER
LNG*: LENGTH

表B. 4. 2 ライトアドレス計算のため

注: 最後のオペレーションの結果は常にアキュムレータ 40 の中に保持される。

【1424】再計算されるべきアドレスがない場合、サイクルカウンタはゼロで空転し、こうしてレジスタのいずれにも書き込まない指令を生じさせる。これは如何なる影響も与えない。

【1425】B. 4. 6. 3 「Bmrecalc」モジュール、Bmrecalcは2クロックサイクル毎に1つのオペレーションを遂行する。それは偶数カウンタサイクル (start alu cyc) 上のbm1 hstt (及びそのバッファと16タイフ) からの指令 50

の中でラッチし、奇数カウンタサイクル (end alu cyc) 上のオペレーション結果をラッチする。オペレーション結果は常に、指令が明記するレジスタに加えて、Accumレジスタの中に記憶される。更に、end alu cyc上で、bmrecalcはたつたいま計算されたアドレスの使用がバッファをフルにするか、エンプティにするかに関して、またいつアドレス及びfull/emptyがうまく計算されるかに関してbmprtlizeに通知する (load addr)。

【1426】Full/emptyはオペレーション結果のサインビットを使用して計算される。

【1427】モジュラスオペレーションは真のモジュラスではないが、 $A \bmod B$ が次のように実施され：
 $(A > B ? (A - B) : A)$

しかしながら、これは次の場合に間違っているだけであり、

$A > (2B - 1)$

これは決して起こらない。

【1428】B. 4. 6. 4 「Bmsnoop」

モジュール、BmsnoopはDRAMインターフェースに供給されるアドレスを監視する4つの18ビットス
 ーパー・スノーパで構成される。スノーパは外部DRA
 Mのオンチップテストを可能にするため、superで
 ななければならない（つまり、クロックラニングでアクセ
 スできなければならない）。これらのスノーパはREQ
 /ACKシステム上で作用しなければならず、従って、
 装置上の他のものとは異なる。

【1429】REQ/ACKは、2線式プロトコールに
 対抗するものとして、このインターフェース上で使用さ
 れる。なぜなら、情報をセnderに送り返す（つまり受
 取を知らせる）ことが必須であり、アクセプトはそれを
 しないからである。従って、これはFIFOポインタを
 厳密に監視する。

【1430】B. 4. 7 「レジスタ」

初期レジスタへのマイクロプロセッサのライトアクセス

を得るために、buf accessに1を書き込むべき
 であり、buf accessが1を読み返す時にアク
 セスが得られる。逆に、マイクロプロセッサのライトア
 クセスを放棄するために、ゼロをbuf access
 に書き込むべきである。buf accessがゼロを
 読み返す時にアクセスが得られる。buf acces
 sは1にリセットされることに注目。

【1431】本発明のダイナミックレジスタ及び初期設
 定レジスタはいつでも読むことができるが、ダイナミッ
 クレジスタがマイクロプロセッサを変化させていないこ
 とを保証するため、ライトアクセスを得なければならない。

【1432】初期設定レジスタは一度だけ書き込まれる
 ことが意図されている。それらを書き直すことにより、
 バッファを間違えて操作させることになるかもしれな
 い。しかしながら、オン・ザ・フライのバッファの長さ
 を増加させ、適時にバッファマネージャに新しい長さを
 使用させることが企図されている。初期設定レジスタ内
 のバリュウが、例えば、バッファがオーバーラップしな
 いことを感じられるかどうかを見るために如何なるチェ
 ックも今まで行われてはいない。これはユーザーの責任
 である。

【1433】

【表208】

レジスタ名	使用法	アドレス
CED BUF ACCESS	xxxxxxxxD	0x24
CED BUF KEYHOLE ADDR	xxDDDDDD	0x25
CED BUF KEYHOLE	DDDDDDDD	0x26
CED BUF CB WR SNP 2	xxxxxxxxDD	0x54
CED BUF CB WR SNP 1	DDDDDDDD	0x55
CED BUF CB WR SNP 0	DDDDDDDD	0x56
CED BUF CB RD SNP 2	xxxxxxxxDD	0x57
CED BUF CB RD SNP 1	DDDDDDDD	0x58
CED BUF CB RD SNP 0	DDDDDDDD	0x59
CED BUF TB WR SNP 2	xxxxxxxxDD	0x5a
CED BUF TB WR SNP 1	DDDDDDDD	0x5b
CED BUF TB WR SNP 0	DDDDDDDD	0x5c
CED BUF TB RD SNP 2	xxxxxxxxDD	0x5d
CED BUF TB RD SNP 1	DDDDDDDD	0x5e
CED BUF CB RD SNP 0	DDDDDDDD	0x5f

表B. 4. 3 バッファマネージャ・ノンキーホールレジスタ

表中、Dがレジスタビットを指示し、xは如何なるレジスタビットも示さない。

[1434]

【表209】

キーホールレジスタ名	使用法	キーホール アドレス
CED BUF CB BASE 3	xxxxxxxx	0x00
CED BUF CB BASE 2	xxxxxxxxDD	0x01
CED BUF CB BASE 1	DDDDDDDD	0x02
CED BUF CB BASE 0	DDDDDDDD	0x03
CED BUF CB LENGTH 3	xxxxxxxx	0x04
CED BUF CB LENGTH 2	xxxxxxxxDD	0x05
CED BUF CB LENGTH 1	DDDDDDDD	0x06
CED BUF CB LENGTH 0	DDDDDDDD	0x07
CED BUF CB READ 3	xxxxxxxx	0x08
CED BUF CB READ 2	xxxxxxxxDD	0x09
CED BUF CB READ 1	DDDDDDDD	0x0a
CED BUF CB READ 0	DDDDDDDD	0x0b

表B. 4. 4 バッファマネージャキーホール内のレジスタ (1/3)

[1435]

【表210】

キーホールレジスタ名	使用法	キーホール アドレス
CED BUF CB NUMBER 3	xxxxxxxx	0x0c
CED BUF CB NUMBER 2	xxxxxxxxDD	0x0d
CED BUF CB NUMBER 1	DDDDDDDD	0x0e
CED BUF CB NUMBER 0	DDDDDDDD	0x0f
CED BUF TB BASE 3	xxxxxxxx	0x10
CED BUF TB BASE 2	xxxxxxxxDD	0x11
CED BUF TB BASE 1	DDDDDDDD	0x12
CED BUF TB BASE 0	DDDDDDDD	0x13
CED BUF TB LENGTH 3	xxxxxxxx	0x14
CED BUF TB LENGTH 2	xxxxxxxxDD	0x15
CED BUF TB LENGTH 1	DDDDDDDD	0x16
CED BUF TB LENGTH 0	DDDDDDDD	0x17

表B. 4. 4 バッファマネージャキーホール内のレジスタ (2/3)

[1436]

【表211】

キーホールレジスタ名	使用法	キーホール アドレス
CED BUF TB READ 3	xxxxxxxxxx	0x18
CED BUF TB READ 2	xxxxxxxxDD	0x19
CED BUF TB READ 1	DDDDDDDD	0x1a
CED BUF TB READ 0	DDDDDDDD	0x1b
CED BUF TB NUMBER 3	xxxxxxxxxx	0x1c
CED BUF TB NUMBER 2	xxxxxxxxDD	0x1d
CED BUF TB NUMBER 1	DDDDDDDD	0x1e
CED BUF TB NUMBER 0	DDDDDDDD	0x1f
CED BUF LIMIT 3	xxxxxxxxxx	0x20
CED BUF LIMIT 2	xxxxxxxxDD	0x21
CED BUF LIMIT 1	DDDDDDDD	0x22
CED BUF LIMIT 0	DDDDDDDD	0x23
CED BUF CSR	xxxxDDDD	0x24

表B. 4. 4 バッファマネージャキーホール内のレジスタ (3/3)

B. 4. 8 「照合」

確認は小さなFIFO'sをダミーのDRAMインターフェースの上に置き、Lsimにおいて、またトップレベルのチップシミュレーションの一部としてC-コードにおいて実施された。

【1437】B. 4. 9 「テスト」

bmanに対するテストカバリッジはbmsnoop内のスノーバ、ダイナミックレジスタ (B. 4. 4 に示した) を通してであり、DRAMインターフェーススキャンチェーンの一部であるスキャンチェーンを用いて行われる。

セクションB. 5 「逆モデラ」

B. 5. 1 「序文」

本文書は本発明による逆モデラ (imodel) 及びトークンフォーマッティング (hsppk) の目的、動作及び実行について説明する。

【1438】注: hsppkはハフマンデコーダの階層型部分であるが、機能的には逆モデラの一部である。従って、本セクションで論じた方がよい。

【1439】B. 5. 2 「展望」

トークンバッファは、imodelとhsppkの間にあり、多量のデータを全てオフチップDRAMに包含することができる。このメモリの効率的な使用を確実にするため、データは16ビットフォーマットでなければならない。トークンフォーマッティングはハフマンデコーダからのデータをトークンバッファのためにこのフォーマットに「バック」する。従って、逆モデラはトークンバッファフォーマットからデータを「取り出す」(アン

バック)。

【1440】しかしながら、逆モデラの主な機能は、「ラン/レベル」コードからゼロデータのランに、そしてレベルに拡張することである。それに加えて、逆モデラはデータトークンが少なくとも64の係数を持ち、スタートアップクライテリアを満たしていないストリームを停止させるための「ゲート」を提供することを保証する。

【1441】B. 5. 3 「インターフェース」

B. 5. 3. 1 「Hsppk」

本発明では、Hsppkは入力としてハフマンデコーダを、そして出力としてトークンバッファを持つ。両インターフェースは2線式タイプのものであり、入力は17ビットのトークンポートであり、出力は16ビットの「バックされたデータ」プラスFLUSH信号である。それに加えて、Hsppkはハフマンクロック発生器からクロックされるので、ハフマンスキャンチェーンに接続される。

B. 5. 3. 2 「Imodel」

Imodelは入力としてトークンバッファスタートアップ出力ゲートロジック (bsogl) を持ち、出力として逆量子化器を持つ。トークンバッファからの入力は16ビットの「バックされたデータ」プラスblockend信号であり、bsoglからの入力は1つのwirestreamenableである。出力は11ビットのトークンポートである。全てのインターフェースは2線式プロトコールにより制御される。Imodelはそれ自体のクロック発生器とスキャンチェーンを持

つ。

【1442】両ブロック共その出力においてスノーバだけに対するマイクロプロセッサアクセスを持つ。

【1443】B. 5. 4 「ブロックの説明」

B. 5. 4. 1 「Hsppk」

Hsppkはハフマンから17ビットデータを受け取り、トークンバッファに16ビットデータを出力する。これはまず入力データを12ビットワードに切り捨てるかスプリットし、次にこれらのワードを16ビットのフォーマットにパックすることにより達成される。

【1444】B. 5. 4. 1. 1 「スプリッティング」

Hsppkは逆ハフマンから17ビットデータを受け取る。このデータは以下のフォーマットを使って17ビットにフォーマットされる。

【1445】Fはフォーマットを指定し；Eは拡張ビット；Rはランビット；Lはレングスビット（サインマグニチュードで示した）あるいはnon-データトークンビット；xはdon't careである。

【1446】

FLLLLLLLLLLLLLFormat 0

ELLLLLLLLLLLLLFormat 0a

FRRRRRRR00000Format 1

通常のトークンは底部12ビットを占めるだけであり、以下の形態を取る：

ExxxxxxLLLLLLLLLLLL

しかしながら、データトークンは以下の形態で各ワードの中にラン及びレベルを持つ：

ERRRRRRLLLLLLLLLLLL

これは次のフォーマットに分けられる：

ERRRRRRLLLLLLLLLLLL->FRRRRR
RR00000

Format 1、ELLLLLLLLLLLLLFormat 0a

10 あるいは、ランがゼロフォーマットであれば、0が使用され：

E000000LLLLLLLLLLLL->FLLLLL
LLLLLLLL

Format 0

フォーマット0において、拡張ビットが失われ、1であると仮定されることが解る。従って、拡張がゼロである場合、それは使用できない。この場合、フォーマット1は無条件に使用できる。

【1447】B. 5. 4. 1. 2 「バッキング」

20 スプリッティングの後、全てのデータワードは12ビット幅である。4つの全ての12ビットワードは3つの16ビットワードに「パック」される：

【1448】

【表212】

入力ワード	出力ワード
000000000000	0000000000001111
111111111111	1111111122222222
222222222222	2222333333333333
333333333333	

表B. 5. 1 バッキング方法

B. 5. 4. 1. 3 「バッファのフラッシング」

本発明のDRAMインターフェースはブロック、32の16ビット「バック済み」ワードを集め、バッファに書き込む。これは、ブロックが部分的にしか完了していない場合、データをストリームの終わりでDRAMインターフェースの中に張り付けることができることを意味している。従って、フラッシングメカニズムが必要である。従って、Hsppkは現在の部分的完了ブロックを無条件に書き込むようにDRAMインターフェースに

信号を送る。

【1449】B. 5. 4. 2. 1 「Imup（アンバッカー）」

40 Imupは3つの機能を果たす：

4) その16ビットのフォーマットからのデータを12ビットワードにアンパックする。

【1450】

【表213】

入力ワード	出力ワード
00000000000001111	00000000000000
1111111122222222	1111111111111
2222333333333333	2222222222222
	3333333333333

表B. 5. 2 アンパッキング方法

5) トークンバッファのフラッシングの間正しいデータを維持する。

【1451】現在の部分的完了ブロックを無条件に書き込むことにより、DRAMインターフェースがフラッシュする時、役に立たないデータはブロックに残ったままである。Imupはブロックの終わりまで、役に立たないデータを、つまりフラッシュトークンからの全てのデータを削除しなければならない。

【1452】6) スタートアップクライテリアが満たされるまで、データを引き留める。ブロックからのデータ出力は、「有効な」(stream enable)が

B. 5. 4. 2. 2 「Imex (EXpander)」

本発明では、Imexは全てのランレングスコードをゼロのランと、それに続くレベルに拡張する。

【1453】B. 5. 4. 2. 3 「Impad (PAlder)」

Impadは全てのデータトークンボディが64 (もしくはそれ以上) のワードを含むことを確実にする。それはトークンの最後のワードをゼロでパディングすることにより行われる。データトークンはボディにおいて64以上のワードを持つためにチェックされない。

【1454】B. 5. 5 「ブロック実行」

B. 5. 5. 1 「Hspk」

典型的に、1サイクルにおいてスプリッティングとパッキングが行われる。

【1455】B. 5. 5. 1. 1 「スプリッティング」

まず、フォーマットが決定される：

IF (datatoken)

IF (lastformat==1) use format 0a;

ELSE IF (run==0) use format 0;

ELSE use format 1;

そしてフォーマットビットが決定される：format

0 format bit = 0;

format 0a format bit = extension bit;

format 1 format bit = 1;

format 1が使用される場合、コードレベルがまだ出力されなければならないので、次のサイクルにおいて新しいデータをアクセプトすべきではない。

B. 5. 5. 1. 2 「パッキング」

パッキング手順は4つの有効なデータ入力毎に循環する。16ビットワードの出力は保持されている最後の有効なワードとそれに続くワードから形成される。これが有効でない場合、出力も有効ではない。手順は以下の通りである：

【1456】

【表214】

	保持されるワード	次に続くワード
有効なサイクル 0	xxxxxxxxxxxx	000000000000
有効なサイクル 1	000000000000	111111111111
有効なサイクル 2	111111111111	222222222222
有効なサイクル 3	222222222222	333333333333

	バック済みワード	
有効なサイクル0	xxxxxxxxxxxx	出力しない
有効なサイクル1	0000000000001111	出力
有効なサイクル2	1111111122222222	出力
有効なサイクル3	2222333333333333	出力

表B. 5. 3 パッキング手順

表中、xは未定義のビットを指示する。

【1457】有効なサイクル0の間、如何なるワードも有効ではないので出力されない。

【1458】有効なサイクルナンバーがリングカウンタによって維持される。それはスプリッタからの有効なデータ及びアクセプトされた出力によって増分される。

【1459】FLUSH (またはpicture end) トークンが受け取られ、トークン自体が出力の準備が整うと、フラッシュ信号がDRAMインターフェース 30 に出力され、有効なサイクルをゼロにリセットする。フラッシュトークンがサイクル3以外のものに到達する

と、トークン自体が出力することを保証するため、フラッシュ信号は有効なサイクルを遅らせなければならない。

【1460】B. 5. 5. 2 「Imodel」

B. 5. 5. 2. 1 「Imup (アンバッカー)」
バッカーの場合と同様に、最後の有効な入力記憶され、次の入力と組み合わせられて、アンパッキングを可能にする。

【1461】

【表215】

	次に続くワード
有効なサイクル0	0000000000001111
有効なサイクル1	1111111122222222
有効なサイクル2	2222333333333333
有効なサイクル3	2222333333333333

	保持されるワード
有効なサイクル0	xxxxxxxxxxxxxxxxxx
有効なサイクル1	0000000000001111
有効なサイクル2	1111111122222222
有効なサイクル3	1111111122222222

	アンパックされるワード	
有効なサイクル0	000000000000	入力
有効なサイクル1	111111111111	入力
有効なサイクル2	222222222222	入力しない
有効なサイクル3	333333333333	入力

表B. 5. 4 アンパッキング手順

表中、xは未定義のビットを指示する。

【1462】有効なサイクルナンバーがリングカウンタによって維持される。アンパック済みデータはトークンのデータ、フラッシュ及びそれからデコードされたピク 30
チャエンドを含む。それに加えて、フォーマット及び拡張ビットがアンパック済みデータからデコードされる。

【1463】formatbit is extn=
(lastformat==1) 11 databo
dy

format=databody && (forma
tbit && lastformatbit)

トークンデコーディングのため、またimexに送られるために。

【1464】FLUSH (またはpicture en 40
d) トークンがアンパックされ、imexに出力され、ブロック・エンド信号がDRAMインターフェースから受け取られるまで、全てのデータが削除される (Validは低くされる)。

【1465】B. 5. 5. 2. 2 「Imex (EX
pander)」

本発明によれば、imexはラン／レベルコードを外へ
拡大するための、4つのステートマシンである。ステ
ートマシンは次の通りである。

・ステート0：ランカウントをランコードからロードする。

【1466】・ステート1：ランカウントを減少させ、
ゼロを出力する。

【1467】・ステート2：入力データ及び出力レ
ベル；デフォルトステート。

【1468】・ステート3：不法なステート。

【1469】B. 5. 5. 2. 3 「Impad (PA
Dder)」

Impadはimexにより、データトークンヘッダに
関して知らされる。次に、それがトークンボディの中の
係数の数をカウントする。64の係数になる前に、トー
クンが終了すると、ゼロ係数がトークン・エンドで挿入
され、それを64係数になるように完了させる。例え
ば、未拡張データヘッダがそれらの後に挿入された64
のゼロ係数を持っている。64以上の係数を持つデータ
トークンはimpadによって影響されない。

【1470】B. 5. 6 「レジスタ」

本発明のimodel及びhsppkはそれらのスノー
バを除き、マイクロプロセッサレジスタを持たない。

【1471】

【表216】

レジスタ名	使用法	アドレス
CED H SNP 2	VAxxxxxx	0x49
CED H SNP 1	DDDDDDDD	0x4a
...
CED H SNP 0	DDDDDDDD	0x4b
CED IM SNP 1	VAExxDDD	0x4c
CED IM SNP 0	DDDDDDDD	0x4d

表B. 5. 5 imodel及びhsppkレジスタ

表中、V=有効ビット；A=アクセプトビット；E=拡張ビット；D=データビット。

【1472】B. 5. 7 「照合」

選択されたストリームはLsimシミュレーションを通して流れる。

【1473】B. 5. 8 「テスト」

入力におけるimodelに対するテストカバリッジはトークンバッファ出力スノーバを通してであり、出力においては、imodel自体のスノーバを通してである。ロジックはimodel自体のスキャンチェーンでカバーされる。

【1474】hsppkの出力はハフマン出力スノーバを通してアクセスできる。ロジックはハフマンスキャンチェーンを通して見ることができる。

セクションB. 6 「バッファスタートアップ」

B. 6. 1 「序文」

本セクションは本発明によるバッファスタートアップの方法及び実行について説明する。

【1475】B. 6. 2 「展望」

ピクチャ・ストリームを円滑にかつ連続的に表示できることを保証するため、デコーディングを開始できる前に、一定量のデータを集めなければならない。これをスタートアップ条件と呼ぶ。コーディングスタンダードはほぼ集められることが必要なデータ量に翻訳され得るVBVディレイを指定する。全てのストリームはそのデータがトークンバッファから進み、デコーディングを可能にする前に、そのスタートアップ条件を満たすことを確実にすることが、「バッファスタートアップ」の目的である。それは、トークンバッファの出力（つまり、逆モデラ）において、概念的なゲート（出力ゲート）によりバッファの中に保持される。そのスタートアップ条件が一度満たされると、このゲートはストリームのために開かれるだけである。

【1476】B. 6. 3 「インターフェース」

Bscentbit（バッファスタートアップビットカウンタ）はデータパスの中にあり、2線式インターフェースにより通信し、マイクロプロセッサに接続される。更

に、それは2線式インターフェースでbsogl（バッファスタートアップ出力ゲートロジック）に分岐する。Bsoglは2線式インターフェースを介して、出力ゲートを実行するimup（逆モデラアンバッカー）を制御する。

B. 6. 4 「ブロック構造」

Bscentbitはスタートコード検出器とコード化データバッファとの間のデータパスにある。この1つのサイクルブロックはブロックを出る有効なデータワードをカウントし、この数をマイクロプロセッサからロードされるであろうスタートアップ条件（またはターゲット）と比較する。ターゲットが満たされると、bsoglが伝えられる。データはbscentbitによって影響されない。Bsoglはbscentbitとimup（逆モデラ内）の間にある。事実上、それはストリームがそれらのターゲットを満たしたことを示すインジケータの列である。列は別の「インジケータ」がimupにアクセプトされる時に、バッファを出るストリーム（つまり、imupでデータストリーム内に受け取られるフラッシュトークン）によって動かされる。列が空であれば（つまり、そのスタートアップターゲットを満たしたバッファにストリームがなければ）、imup内のストリームは立ち往生させられる。

【1477】その列は有限の深さを持つだけであるが、これはbsoglの中の列を壊すことによって無限に拡大でき、マイクロプロセッサがその列を監視できるようにする。これらの列のメカニズムは内部列及び外部列と各々称される。

【1478】B. 6. 5 「ブロック実行」

B. 6. 5. 1 「Bscentbit（バッファスタートアップビットカウンタ）」

Bscentbitはバッファスタートアップに入力される全ての有効なワードをカウントする。カウンタ（bsctr）は16～24ビット幅のプログラム可能カウンタである。更に、bsctrはそれに十分な速度を与えるためにキャリールックアヘッド回路を具備する。Bsctrの幅はcedbsprescaleによって

457

プログラムされる。それはビットを8~16高くすることにより行われ、それにより常にキャリアが送られるようになる。従って、それらのビットは効果的に使用されないままである。bsctrの上位8ビットがターゲット(ced bs target)との比較のために使用される。

【1479】比較(ced bs count \geq ced bs target)がbscmpにより行われる。

【1480】ターゲットはストリームがハフマンデコーダの中にあり、マイクロプロセッサにより計算される時に、ストリームから引き出される。従って、それはストリーム・スタート後に設定されるだけであろう。スタートアップ前に、target validは低く設定される。ced bs targetへの書き込みはtarget validを高く設定し、bscmpにおける比較が行われるようにする。比較がced bs count \geq ced bs targetであることを示すと、target validは低く設定される。ターゲットは満たされた。

【1481】ターゲットが満たされると、カウントがリセットされる。それはストリーム・エンドでリセットしないことに注意。それに加えて、それがストリーム・エンドの前であれば、カウンティングはターゲットが満たされた後に不能化される。カウントは255で飽和する。

【1482】ストリーム・エンド(つまり、フラッシュ)がbsbitcntにおいて検出されると、abs flush eventが作られる。ターゲットが満たされる前にストリームが終了すると、追加イベント(bs flush before target met event)が作られる。これらのイベントが発生すると、ブロックは立ち往生させられる。これにより、ユーザーは次のストリームのターゲットサーチをやり直すことができ、あるいはbs flush before target met eventイベントの場合、次のいずれかである：

- 1) target metを強いるであろうゼロのターゲットを書き込むあるいは
- 2) ターゲットが満たされず、最後のストリームと組み合わせられたこれがターゲットに達するまで、次のストリー

458

ームを進ませることに注目。この次のストリームのためのターゲットはそれに応じて調整されるべきである。

【1483】B. 6. 5. 2 「BSOGL (バッファスタートアップ出力ゲートロジック)」

前述したように、Bsoglはストリームがそのターゲットを満たしたことを指示するインジケータ列である。列タイプはced bs queue (内部(0)または外部(1))によって設定される。これは内部列を選択するためのリセットである。列の深さはコード化データバッファ、ハフマン及びトークンバッファにおいて存在できる満たされたストリームの最大数を決定する。この数に達すると(つまり、列が一杯になると)、bsoglはデータバスをbsbitcntで立ち往生させる。

【1484】内部列の使用はマイクロプロセッサからの如何なる動作も必要としない。しかしながら、列の深さを増加させる必要があれば、(設定されるべきced bs queueへのアクセスを得るためにced bs accessを設定し、target met event及びstream end eventが可能化され、アクセスが放棄されることにより)外部列を設定できる。

【1485】外部列(マイクロプロセッサにより維持されるカウント)は内部列に挿入される。外部列は2つのイベント: target met event及びstream end eventにより維持される。これらは単に各々service queue input、service queue output、及びレジスタced bs enable next streamと称される。事実上、target met eventは列を供給する内部列のアップストリーム・エンドである。同様に、ced bs enable next streamは列を消費する内部列のダウンストリーム・エンドである。同様に、stream end eventはダウンストリーム列を供給するためのリクエストであり; stream end eventはced bs enable next streamをリセットする。2つのイベントは次のようにサービスされるべきである：

```
/*TARGET MET EVENT*/
j = micro read (CED BS ENABLE NXT STM
):
if (j == 0) /*Is next stream enabled?*/
(/*no, enable it*/
micro write
(CED BS ENABLE NXT STM, 1);
printf("enable next stream (queue
=0x%x)\n", (context->queue));
```

459

460

```

else /*yes, increment the queue of
      " target met" streams*/
{
    queue++;
    printf(" stream already enabled
           (queue=0x%x) \n", (context->queue));
}
/*STREAM EVENT*/
if (queue>0) /*are there any
              " target mets" left?*/
{
    /*yes, decrement the queue and
      enable another stream*/queue--;
    micro write
      (CED BS ENABLE NXT STM, 1);
    printf(" enable next stream (queue
           =0x%x) \n", (context->queue));
}
else
    printf(" queue empty cannot enable
           next stream (queue=0x%x) \
           n", queue);
    micro write (CED EVENT 1, 1 << BS
      STREAM END EVENT); /*clear event*/

```

列タイプは随時（上述の手段により）内部から外部へと変更することができるが、外部列が（上記" queue ==0から）空である時は、リセットされるべきced bs queueへのアクセスを得るため、ced bs accessを設定し、target met event及びstream endeventをマスクし、アクセスを放棄することにより、外部から内部へ

エックを不能にし、ced bsqueue（外部）を設定し、target met event及びstream end eventをマスクし、ced bs enable nxtstreamを設定する。この方法で、全てのストリームが常に可能化される。B.

6. 6 「マイクロプロセッサレジスタ」

[1487]

[表217]

[1486] 他方、ストリームスタートアップ条件のチ

レジスタ名	使用法	アドレス
CED BS ACCESS	xxxxxxxxD	0x10
CED BS PRESCALE	xxxxxDDD	0x11
CED BS TARGET	DDDDDDDD	0x12
CED BS COUNT	DDDDDDDD	0x13
BS FLUSH EVENT	rrrrrDrr	0x02
BS FLUSH MASK	rrrrrDrr	0x03
BS FLUSH BEFORE TARGET MET EVENT	rrrrDrrr	0x02
BS FLUSH BEFORE TARGET MET MASK	rrrrDrrr	0x03

表B. 6. 1 Bscntbltレジスタ

[1488]

50 [表218]

レジスタ名	使用法	アドレス
TARGET MET EVENT	rrrDrrrr	0x02
TARGET MET MASK	rrrDrrrr	0x03
STREAM END EVENT	rrDrrrrr	0x02
STREAM END MASK	rrDrrrrr	0x03
CED BS QUEUE*	xxxxxxxxD	0x14
CED BS ENABLE NXT STM*	xxxxxxxxD	0x15

表B. 6. 2 Bsogレジスタ
レジスタ名、使用法、アドレス

表中、

- ・Dはレジスタビットであり、
- ・xは存在しないレジスタビットであり、
- ・rは予約済みレジスタビットであり、
- ・これらのレジスタへのアクセスを得るため、割り込みサービスルーチンの場合でなければ、ced bs accessは1に設定され、それが1を読み戻すまで登録されなければならない。ced bs accessをゼロに設定することによりアクセスが放棄される。

【1489】セクションB. 7 「DRAMインターフェース」

B. 7. 1 「展望」

本発明において、空間デコーダ、時間デコーダ及びビデオフォーマッティング部は各々その特別なチップのためにDRAMインターフェースブロックを含む。3つの装置全てにおいて、DRAMインターフェースの機能は、チップから外部DRAMへのデータ伝送、及びアドレス発生器により供給されるブロックアドレスを介して外部DRAMからチップへのデータ伝送である。

【1490】DRAMインターフェースは典型的に、アドレス発生器に対して、またそれを通してデータが送られる様々なブロックのクロックに対して非同期であるクロックから操作する。しかしながら、クロックはほぼ同じ周期で操作するので、この非同期性は容易に処理できる。

【1491】データは通常DRAMインターフェースと64バイトのブロック内の残りのチップとの間に伝送される(唯一の例外は時間デコーダにおける予測データである)。伝送は「スイングバッファ」として知られる装置によって発生する。これは本質的にダブルバッファード構成において操作される一対のDRAMであり、DRAMインターフェースが1つのRAMを満たしたり、あるいは空にする一方、チップの別の部分が他のRAMを空にしたり、満たしたりする。アドレス発生器からアドレスを運ぶ別のバスが各スイングバッファと連合する。

【1492】各チップは4つのスイングバッファを持つ

が、これらのスイングバッファの機能は各々の場合により異なる。空間デコーダの場合、1つのスイングバッファがコード化データをDRAMに伝送するために使用され、別のスイングバッファがDRAMからコード化データを読むために使用され、三番目のスイングバッファがトークン化されたデータをDRAMに伝送するために使用され、四番目のスイングバッファがDRAMからトークン化されたデータを読むために使用される。時間デコーダにおいては、1つのスイングバッファがイントラもしくは予測されたピクチャデータをDRAMに書き込むために使用され、二番目のバッファがDRAMからイントラもしくは予測されたピクチャデータを読むために使用され、他の2つのバッファが予測データを前方及び後方に読むために使用される。ビデオフォーマッティング部においては、1つのスイングバッファがデータをDRAMに伝送するために使用され、他の3つのスイングバッファがDRAMからデータを読むために使用され、各々がルミナンス(Y)、赤及び青の色差データ(各々Cr及びCb)のためである。

【1493】以下のセクションは本発明によるDRAMインターフェースのオペレーションを説明し、それは空間デコーダDRAMインターフェースのオペレーションと本質的に同じである、1つのライトスイングバッファと1つのリードスイングバッファを持っている。これは図140の「DRAMインターフェース」に図示している。

【1494】B. 7. 2 「一般的なDRAMインターフェース」

図140において、アドレス発生器に対するインターフェース、及びデータを供給し、データを取るブロックに対するインターフェースは全て2線式インターフェースである。アドレス発生器は制御トークンの受取の結果としてアドレスを発生させるか、あるいは単にアドレスの固定シーケンスを発生させることができる。DRAMインターフェースは特別の方法で、アドレス発生器と連合する2線式インターフェースを処理する。アドレスを受

け取る準備ができた時にアクセプトラインを高く保持する代わりに、DRAMインターフェースはアドレス発生器が有効なアドレスを供給し、そのアドレスを処理し、1つのクロック周期の間アクセプトラインを高く設定する。こうして、リクエスト/アクノレッジ (REQ/A CK) プロトコルを実行する。

【1495】DRAMインターフェースの独得の特徴は、アドレス発生器及び他のブロックとは完全に別個にデータを提供/アクセプトするブロックと連絡する能力である。例えば、アドレス発生器はライトスイングバッファにおいてデータと関連するアドレスを発生させることができるが、ライトスイングバッファが外部DRAMに書き込まれる用意が整ったデータブロックがあることを合図しないとどのような行動も取らないであろう。しかしながら、アドレス発生器からの適当なバスにアドレスが供給されない限り、どのような行動も取られない。更に、ライトスイングバッファ内のRAMの1つが一度データで満たされると、他のRAMもデータ入力が立ち往生させられる(2線式インターフェースアクセプト信号が低く設定される)前に、完全に満たされ、DRAM 20 インターフェースに「スイング」される。

【1496】本発明のDRAMインターフェースのオペレーションを理解する際に注目すべき重要なことは、適切に構成されたシステムにおいて、少なくともスイングバッファと残りのチップ間の全ての平均データ速度の合計と同程度に速く、DRAMインターフェースがスイングバッファと外部DRAMの間にデータを伝送することができることである。

【1497】各DRAMインターフェースは次にサービスするのはどのスイングバッファかを決定する方法を含む。30 一般に、これは「ラウンドロビン」であるか、もしくはプライオリティエンコーダのいずれかであり、ラウンドロビンの場合、サービスされるスイングバッファが最近めつたに順番が回ってこなかった次に利用できるスイングバッファであり、プライオリティエンコーダの場合、いくつかのスイングバッファが他のバッファより高い優先順位を持つ。両方の場合において、他の全てのリクエストより高い優先順位を持つリフレッシュリクエスト発生器から追加リクエストが来るであろう。リフレッシュリクエストはマイクロプロセッサインターフェース 40 を介してプログラム可能なリフレッシュカウンタから発生される。

【1498】B. 7. 2. 1 「スイングバッファ」
図141はライトスイングバッファを示す。オペレーションは次の通りである：

1) 有効なデータは入力 (data in) において表示される。各データ片がアクセプトされるにつれて、それはRAM1に書き込まれ、アドレスが増分される。

【1499】2) RAM1が一杯である時、入力データは制御をきり、RAM1が読み出される準備ができ 50

たことを指示するためリードサイドに信号を送る。この信号は2つの非同期クロックレジームの間を通り、従って3つの同期フリップフロップを通過する。

【1500】3) 入力サイドに到着すべき次のデータアイテムは、まだ空であるRAM2に書き込まれる。

【1501】4) ラウンドロビンもしくはプライオリティエンコーダが、このスイングバッファを読む番であることを指示すると、DRAMインターフェースはRAM1の内容を読み、それらを外部DRAMに書き込む。そうすれば、(2)と同様に、信号が非同期インターフェースを横切って送り返され、RAM1にもう一度書き込む準備ができたことを指示する。

【1502】5) DRAMインターフェースがRAM1を空にし、入力サイドがRAM2を満たす前にそれを「スイング」すれば、データは連続的にスイングバッファによりアクセプトされることができ、そうでなければ、RAM2が満たされた時、RAM1が入力サイドにより使用されるために「スイングバック」されるまで、スイングバッファはそのアクセプト信号を低く設定するであろう。

【1503】6) このプロセスは無限に繰り返される。

【1504】リードスイングバッファのオペレーションも同様であるが、入力と出力のデータバスが逆である。

【1505】B. 7. 2. 2 「外部DRAM及びスイングバッファのアドレッシング」

DRAMインターフェースは利用可能なメモリー帯域幅を最大にするように設計される。従って、各8×8データブロックが同じDRAMページに記憶されるように配置される。この方法で、DRAM高速ページアクセスモードを完全に使用でき、その場合1つのローアドレスが供給され、続いて多くのカラムアドレスが供給される。それに加えて、外部DRAMに対するデータバスが8、16、または32ビット幅にでき、使用されるDRAM量が特別なアプリケーションのサイズ及び帯域幅の要件に適合できるようにするための便宜が提供される。

【1506】(空間デコーダ上のDRAMインターフェースがどのように作用するかを示す) この例では、アドレス発生器は、リードスイングバッファとライトスイングバッファの各々のために、DRAMインターフェースにブロックアドレスを提供する。このアドレスはDRAM用のローアドレスとして使用される。カラムアドレスの6ビットはDRAMインターフェース自体により供給され、これらのビットは更にスイングバッファRAM用のアドレスとしても使用される。スイングバッファに対するデータバスは32ビット幅であり、従って、外部DRAMに対するバス幅が32ビットより狭い場合、次のワードがライトスイングバッファから読まれる前に、あるいは次のワードがリードスイングバッファに書き込まれる前に(リード及びライトは外部DRAMに対する伝送方向に關係する)、2つから4つの外部DRAMアク

セスが行われなければならない。

【1507】時間デコーダ及びビデオフォーマッティング部の場合は、状況はもっと複雑である。これらについては下記において別に説明する。

【1508】B. 7. 3 「DRAMインターフェースタイミング」

本発明において、DRAMインターフェースタイミングブロックは、DRAM信号のエッジをシステムクロック周期の正確に $1/4$ に置くためにタイミングチェーンを使用する。フェイズロックループからの2つの直角クロックが使用される。これらは概念的な $2x$ クロックを形成するために組み合わせられる。いずれか1つのチェーンが、 $2x$ クロックの反対のフェイズ上で並列する2つのシフトレジスタから作られる。

【1509】まず最初に、ページスタートサイクルのために1つのフェイズがあり、リード/ライト/リフレッシュサイクルのために別のフェイズがある。各サイクルの長さはマイクロプロセッサインターフェースを介してプログラム可能であり、その後ページスタートチェーンが固定長を持ち、サイクルチェーンの長さはページスタート中に適当に変化する。

【1510】リセットされると、チェーンはクリアされ、パルスが作られる。このパルスはチェーンに沿って移動し、DRAMインターフェースからのステート情報により方向付けられる。DRAMインターフェースクロックはこのパルスにより発生される。各DRAMインターフェースクロック周期はDRAMの1サイクルに対応

する。このように、DRAMサイクルが異なる長さを持つので、DRAMインターフェースクロックは一定速度ではない。

【1511】更に、タイミングチェーンは上記チェーンからのパルスをDRAMインターフェースからの情報と組合せ、出力ストロブ及びイネーブル(notcas, notras, notwe, notoe)を発生させる。

【1512】セクションB. 8 「逆量子化器」

B. 8. 1 「序文」

本文書は本発明による逆量子化器(iq)の目的、動作及び実行について説明する。

【1513】B. 8. 2 「展望」

逆量子化器は量子化された係数からの係数、量子化重量、及びステップサイズを再構築し、その全てがデータストリーム内で送信される。

【1514】B. 8. 3 「インターフェース」

Iqはデータバスにおいて逆モデラと逆DCTの間であり、マイクロプロセッサに接続される。データバス接続は2線式インターフェースを介して行われる。入力データは10ビット幅であり、出力データは11ビット幅である。

【1515】B. 8. 4 「逆量子化器の演算」

B. 8. 4. 1 「H261式」

【1516】

【数1】

イントラモードにおいてコード化されるブロックのために：

$$\begin{aligned}
 \dot{C}_i &= 8Q_i \quad i = 0 \\
 \dot{C}_i &= \text{iq_quant_scale}(2Q_i + \text{sign}(Q_i)) \\
 \dot{C}_i &= \dot{C}_i - \text{sign}(\dot{C}_i) \quad \dot{C}_i = \text{even} \\
 \dot{C}_i &= \dot{C}_i \quad \dot{C}_i = \text{odd} \quad \left. \begin{array}{l} \\ \end{array} \right\} 0 < i < 64 \\
 C_i &= \min(\max(\dot{C}_i, -2048), 2047)
 \end{aligned}$$

他の全てのコード化されるブロックのために：

$$\begin{aligned}
 \dot{C}_i &= \text{iq_quant_scale}(2Q_i + \text{sign}(Q_i)) \\
 \dot{C}_i &= \dot{C}_i - \text{sign}(\dot{C}_i) \quad \dot{C}_i = \text{even} \\
 \dot{C}_i &= \dot{C}_i \quad \dot{C}_i = \text{odd} \quad \left. \begin{array}{l} \\ \end{array} \right\} 0 \leq i < 64 \\
 C_i &= \min(\max(\dot{C}_i, -2048), 2047)
 \end{aligned}$$

$$\begin{aligned}
 \dot{C}_i &= \left\lfloor \frac{\text{iq_quant_scale}(W_{i,j})(2Q_i + \text{sign}(Q_i))}{16} \right\rfloor \\
 \dot{C}_i &= \dot{C}_i - \text{sign}(\dot{C}_i) \quad \dot{C}_i = \text{even} \\
 \dot{C}_i &= \dot{C}_i \quad \dot{C}_i = \text{odd} \quad \left. \begin{array}{l} \\ \end{array} \right\} \begin{array}{l} 0 < i < 64 \\ j = 1, 3 \end{array} \\
 C_i &= \min(\max(\dot{C}_i, -2048), 2047)
 \end{aligned}$$

B. 8. 4. 2 「JPEG式」

[1517]

【数2】

$$\dot{C}_i = W_{i,j}Q_i + 1024 \quad i = 0$$

$$\dot{C}_i = W_{i,j}Q_i \quad 0 < i < 64$$

$$C_i = \min(\max(\dot{C}_i, -2048), 2047)$$

$$j = \text{jpeg_table_indirection}(c)$$

B. 8. 4. 3 「MPEG式」

[1518]

【数3】

イントラモードにおいてコード化されるブロックのために:

$$\begin{aligned}
 \hat{C}_i &= W_{i,j} Q_i + 1024 \quad i = 0 \\
 \hat{C}_i &= \text{floor} \left(\frac{2iq_quant_scale W_{i,j} Q_i}{16} \right) \\
 \left. \begin{aligned} \hat{C}_i &= \hat{C}_i - \text{sign}(\hat{C}_i) & \hat{C}_i &= \text{even} \\ \hat{C}_i &= \hat{C}_i & \hat{C}_i &= \text{odd} \end{aligned} \right\} \quad \begin{aligned} 0 &< i < 64 \\ j &= 0, 2 \end{aligned} \\
 C_i &= \min(\max(\hat{C}_i, -2048), 2047)
 \end{aligned}$$

ゼロにリセットされるハフマンにおいて、予測者に説明するためのイントラD
Cの場合に1024が加算される。

他の全てのコード化されるブロックのために:

$$\begin{aligned}
 \hat{C}_i &= W_{i,j} Q_i + 1024 \quad i = 0 \\
 \hat{C}_i &= \text{floor} \left(\frac{2iq_quant_scale W_{i,j} Q_i}{16} \right) \\
 \left. \begin{aligned} \hat{C}_i &= \hat{C}_i - \text{sign}(\hat{C}_i) & \hat{C}_i &= \text{even} \\ \hat{C}_i &= \hat{C}_i & \hat{C}_i &= \text{odd} \end{aligned} \right\} \quad \begin{aligned} 0 &< i < 64 \\ j &= 0, 2 \end{aligned} \\
 C_i &= \min(\max(\hat{C}_i, -2048), 2047)
 \end{aligned}$$

B. 8. 4. 4 「J PEGバリエーション式」

[1519]

[数4]

$$\begin{aligned}
 \hat{C}_i &= \text{floor} \left(\frac{2iq_quant_scale W_{i,j} Q_i}{16} \right) + 1024 \quad i = 0 \\
 \hat{C}_i &= \text{floor} \left(\frac{2iq_quant_scale W_{i,j} Q_i}{16} \right) \quad 0 < i < 64 \\
 C_i &= \min(\max(\hat{C}_i, -2048), 2047) \\
 j &= \text{jpeg_table_indirection}(c)
 \end{aligned}$$

B. 8. 4. 5 「他の全てのトークン」

データトークン以外の全てのトークンは量子化されない
iqを通過しなければならない。そこで:

$$\begin{aligned}
 \text{sign}(a) &= -1 & a < 0, \\
 &= 0 & a = 0, \\
 &= 1 & a > 0
 \end{aligned}$$

$$\begin{aligned}
 \max(a, b) &= a & a > b, \\
 &= b & a \leq b
 \end{aligned}$$

$$\begin{aligned}
 \min(a, b) &= a & a \leq b, \\
 &= b & a > b
 \end{aligned}$$

Floor(a) は整数を次のように戻す:

$$\begin{aligned}
 (a-1) &< \text{floor}(a) \leq a & a \geq 0, \\
 a &\leq \text{floor}(a) < (a+1) & a \leq 0
 \end{aligned}$$

Qiは量子化係数である。

[1520] Ciは再構築された係数である。

[1521] Wi,jは量子化テーブルマトリックスにお
ける値である。

[1522] iはジグザグに沿った係数インデックスで
30 ある。

[1523] jは量子化テーブルマトリックスナンバー
(0 ≤ j ≤ 3) である。

[1524] B. 8. 4. 6 「結合されるマルチプル
スタンダード」

上記スタンダード及びそれらのバリエーションの全て
(更にiqによって変更されてはならない制御データ)
は1つの式にマップできる:

$$\text{OUTPUT} = (2 \text{ INPUT} + k) (xy) / 16$$

以下の追加ポスト逆量子化機能がある:

・1024を加算する

・サインマグニチュードから2の補数表示への変換

・全ての偶数をゼロに向かって最も近い奇数にラウンド
する

・結果を+2047または-2048に飽和させる。

[1525] スタンダードの各バリエーションのための
変数k, x及びy, 及びそれらが使用するその関数は表
219, 表220に示す。

[1526] B. 8. 4. 6 「結合されるマルチプル
スタンダード」

[1527]

【表219】

スタンダード		x	y	k
		重量	スケール	
H261	イントラDC	8	8	0
	イントラ	16	iq quant scale	1
	その他	16	iq quant scale	1
JPEG	DC	W _i	8	0
	その他	W _i	8	0
MPEG	イントラDC	8	8	0
	イントラ	W _i	iq quant scale	0
	その他	W _i	iq quant scale	1
XXX	DC	W _i	iq quant scale	0
	その他	W _i	iq quant scale	0
他のトークン		1	8	0

表B. 8. 1 制御デコーディング (1/2)

【1528】

【表220】

スタンダード		加算	ラウンド	サデュレーション	変換
		1024	偶数	残	2の補数
H261	イントラDC	No	No	Yes	Yes
	イントラ	No	Yes	Yes	Yes
	その他	No	Yes	Yes	Yes
JPEG	DC	Yes	No	Yes	Yes
	その他	No	No	Yes	Yes
MPEG	イントラDC	Yes	No	Yes	Yes
	イントラ	No	No	Yes	Yes
	その他	No	Yes	Yes	Yes
XXX	DC	Yes	No	Yes	Yes
	その他	No	No	Yes	Yes
他のトークン		No	No	No	No

表B. 8. 1 制御デコーディング (2/2)

B. 8. 5 「ブロック構造」

段落B. 8. 4. 6及び表220から、マルチスタンダード逆量子化器用に1つの構造を使用できることが解る。その算術的ブロック線図を図142「算術的ブロック」に示す：算術的ブロック用の制御は2つのセクションに機能的に分割することができる：

・ステータスレジスタまたは量子化テーブルにロードするためのトークンのデコーディング、
・ステータスレジスタの制御信号へのデコーディング。

【1529】トークンは次のサイクル、つまりレジスタ 10 のiqcbのバンクを制御するiqcaにおいてデコードされる。それは更にigram内の4つの量子化テーブルへのアクセスを制御する。算術、つまり、2つの乗

```

if (tokenheader==QUANT SCALE)
{
    sprintf(preport, "QUANT SCALE");
    reg addr=ADDR IQ QUANT SCALE;
    rnotw=WRITE;
    enable=1;
}
if (tokenheader==QUANT TABLE)
    /*QUANT TABLE token*/
switch (substate)
{
    case 0: /*quantisation table header*/
        sprintf(preport,
            "QUANT TABLE %s s0",
            (headerextn ? "(full)" :
                "(empty)"));
        nextsubstate=1;
        insertnext=(headerextn ? 0:1);
        reg addr=ADDR IQ COMPONENT;
        rnotw=WRITE;
        enable=1;
        break;
    case 1: /*quantisation table body*/
        sprintf(preport,
            "QUANT TABLE %s s1",
            (headerextn ? "(full)" :
                "(empty)"));
        nextsubstate=1;
        insertnext=(headerextn ?
            0 : (qtm addr 63==0));
        reg addr=USE QTM;
        rnotw=(headerextn ? WRITE : READ);
        enable=1;
        break;
    default:
        sprintf(preport, "ERROR in iq

```

算器とポスト関数はiqarithにある。iq用の完全なブロック線図を図143に示す。

【1530】B. 8. 6 「ブロック実行」

B. 8. 6. 1 「Iqca」

発明において、iqcaはトークンをigram及びiqcb内のレジスタ用制御信号にデコードするために使用されるステートマシンである。ステートマシンは各々の新しいトークンによってリセットされるので、各トークン用のステートマシンとして説明した方がよ

い。例えば：QUANT SCALE (B. 8. 7. 4, QUANT SCALEを参照) 及びQUANT TABLE (B. 8. 7. 6 QUANT TABLEを参照) 用のコードは以下の通りである：

```

quantisation table tokendecoder
(substate %x) \n,
substate);
break;
}
}

```

サブステートがトークン内のステートである場合、QUANT SCALEは、例えば1つのサブステートだけを持つ。しかしながら、QUANT TABLEは2つのサブステートを持ち、1つはヘッダであり、他の1つはトークンボディである。

【1531】ステートマシンはPLAとして実装される。未認識トークンは如何なるワードラインも発生させないし、PLAにもデフォルト（無害な）制御を出力させない。

【1532】従って、iqcaはBodyWordカウンタからアドレスをigramに供給し、例えば未拡張QUANT TABLE (B. 8. 7. 4を参照) において、ワードをストリームに挿入する。これは出力を有効に保つ一方で、入力を立ち往生させることによって達成される。続くブロック (iqcbまたはiqarith) において正しいデータでワードを満たすことができる。

【1533】iqcaは2線式インターフェースによって制御されるデータバス内の1サイクルである。

【1534】B. 8. 6. 2 「iqcb」
発明において、iqcbはiqステータスレジスタを保持する。iqcaの制御下、iqcbはこれらをデータバスから／へとロードもしくはアンロードする。

【1535】ステータスレジスタは、XY乗算器ターム及びポスト量子化機能を制御するため、iqarithのために制御ワイヤにデコードされる（表219、表220を参照）。

【1536】データバスのサインビットはここで分離され、ポスト量子化機能に送られる。更に、データバス上のゼロバリューワードがここで検出される。その後算術は無視され、ゼロがデータバスにマックスされる。これはiqの「ゼロイン；ゼロアウト」スペックに従う最も簡単な方法である。

【1537】ステータスレジスタは、レジスタiqaccessが1に設定され、1を読み戻す時にのみ、マイクロプロセッサからアクセス可能である。この状況では、iqcbはデータバスを停止させ、こうしてレジスタが安定したバリューを持ち、如何なるデータもデータバスにおいて転換されない。

【1538】Iqcbは2線式インターフェースによって制御されるデータバス内の1サイクルである。

【1539】B. 8. 6. 3 「Iqram」
Iqramは各々が648ビットである4つの量子化テーブルマトリクス(QTM)に対して支持しなければ

ならない。従って、それはサイクル毎に1リードまたは1ライトが可能な、2568ビットの6トランジスタRAMである。RAMはその制御を受け取る2線式インターフェースロジックにより囲まれ、iqcaからデータを書き込む。それはiqarithにデータを読み出す。同様に、igramはiqcbと同じデータバス内のサイクルを占める。

【1540】RAMはiqaccessが1を読み返す時、マイクロプロセッサから読み出され、書き込まれる。RAMはキーホールレジスタ、iqqtmkeyholeの背後に置かれ、iqqtmkeyholeaddrによってアドレスされる。iqqtmkeyholeへのアクセスはiqqtmkeyholeaddr内に保持される、それが指すアドレスを増分させるであろう。同様に、iqqtmkeyholeaddrは直接書き込むことができる。

【1541】B. 8. 6. 4 「iqarith」
iqarithは3つのサイクルに亘ってパイプラインでつながれ、スプリットされる3つの機能があることに注意。機能については下記において論じる（図140を参照）。

【1542】B. 8. 6. 4. 1 「XY乗算器」
これはデータバス乗算器に供給される5 (X) x 8 (Y) ビットのキャリセーブ未サイン乗算器である。乗数及び被乗数はiqcbからの制御ワイヤで選択される。乗法は第1サイクルにあり、分解アダーは第2サイクルにある。

【1543】乗算器への入力において、igramからのデータはQUANT TABLEをデータバスに読み出すために、データバスにマックスされ得る。

【1544】B. 8. 6. 4. 2 「(XY) * データバス乗算器」

この13 (XY) x 12 (データバス) ビットのキャリセーブ未サイン乗算器はブロックの3サイクルに亘ってスプリットされる。第1サイクルに3つの部分積、第2サイクルに7つ、第3サイクルに残りの2つである。

【1545】乗算器からの全ての出力は2047以下 (non coefficient) であるか、あるいは+2047/-2048に飽和されるので、上位12ビットは分解される必要がない。従って、分解アダーはたった2ビット幅である。高いオーダーのビットの残りに関して、ゼロ検出がサチュレーション信号として充分である。

【1546】B. 8. 6. 4. 8 「ポスト量子化機

能」

ポスト量子化機能は、

- ・1024を加算する
- ・サインマグニチュードから2の補数表示に変換する
- ・全ての偶数をゼロに向かって最も近い奇数にラウンドする
- ・結果を+2047または-2048に飽和させる
- ・出力をゼロに設定する (B. 8. 6. 2を参照)。

【1547】最初の3つの機能は12ビットアダーで実行される (第2と第3のサイクルにパイプラインでつながれる)。このことから、各機能が必要とするものが何であるかが解り、これらは1つのアダーの上に結合される。

【1548】

【表221】

機能	データバス>0の場合	データバス>0の場合
2の補数に変換	何もしない	1を加算を逆転
全ての偶数をラウンド	1を引く	1を加算
1024を加算	1024を加算	1024を加算

表B. 8. 2 ポスト量子化アダー機能

当業者なら認識するであろうが、これらの機能は結合された時に互いに依存し合うので、これらを再プログラミングする際に注意しなければならない。

【1549】サチュレーションバリュ、ゼロ及びゼロ+1024は第3のサイクルの終わりでデータバスにマックスされる。

【1550】B. 8. 7 「逆量子化器トークン」

以下の説明は逆量子化器が応答する各トークンtpのための、逆量子化器の行為を定義する。全ての場合に、トークンは逆量子化器の出力に送られる。ほとんどの場合、トークンは下記に記す例外を除き、逆量子化器によって修正されない。全ての未認識トークンは逆量子化器の出力に未修正のまま送られる。

【1551】B. 8. 7. 1 「シーケンススタート」このトークンはレジスタiq prediction mode [1:0] 及びiq mpeg indirection [1:0] がゼロに設定されるようにする。

【1552】B. 8. 7. 2 「コーディングスタンダード」

このトークンはデコードされる現在のスタンダード (MPEG、JPEGまたはH. 261) に基づいて、iq standard [1:0] に適切なバリュがロードされるようにする。

【1553】B. 8. 7. 3 「予測モード」

このトークンはiq prediction mode [1:0] をロードする。予測モードトークンは2つ以上のビットを所有するが、逆量子化器は2つの最低オーダーのビットへのアクセスを必要とするだけである。これらはブロックがイントラコード化されているか否かを判断する。

【1554】B. 8. 7. 4 「量子化スケール」

このトークンはid quant scale [4:0] をロードする。

【1555】B. 8. 7. 5 データ

本発明では、このトークンは実際の量子化係数を所有する。トークンヘッドは色成分を識別する2つのビットを含み、これらはiq component [1:0] にロードされる。次の64のトークンワードは量子化係数を含む。これらは逆量子化プロセスの結果として修正され、再構築された係数と置き換えられる。

【1556】正確に64の拡張ワードがトークン内に存在しない場合、逆量子化器の行為は限定されない。

【1557】逆量子化器の入力におけるデータトークンは量子化係数を所有する。これらはサイン・マグニチュードフォーマットにおいて11ビット (10ビット+サインビット) で表示される。バリュ「マイナスゼロ」は使用されるべきではないが、正確にゼロと解釈される。

【1558】逆量子化器の入力におけるデータトークンは再構築された係数を所有する。これらは2の補数フォーマットにおいて12ビット (11ビット+サインビット) で表示される。入力におけるデータトークンは、逆量子化器の入力において持ったのと同じ数のトークン拡張ワードを持つであろう。

【1559】B. 8. 7. 6 「量子化テーブル (QUANT TABLE)」

このトークンは新しい量子化テーブルをロードするため、あるいは現在のテーブルを読み出すために使用できる。逆量子化器においては、典型的に、トークンはビットストリームからデコードされた新しいテーブルをロードするために使用されるであろう。現在のテーブルを読み出す動作は、そのテーブルがビットストリームにおいてエンコードされることになる場合、エンコードの前方の量子化器において有用である。

【1560】トークンヘッドは使用されることになるテーブルナンバーを特定する2つのビットを含む。これら

は `iq component [1:0]` の中に置かれる。このレジスタは今では色成分ではなく、「テーブルナンバー」を含んでいることに注意。

【1561】トークンヘッ드의拡張ビットが1である場合、逆量子化器はそこに正確に64の拡張トークンワードがあることを期待する。各々が量子化テーブルバリューとして解釈され、ロケーションゼロで始まる適切なテーブルの連続するロケーションの中に置かれる。各拡張トークンワードの9番目のビットが無視される。トークンは更に通常の方法で、未修正のまま、逆量子化器の出力にも送られる。

【1562】トークンヘッ드의拡張ビットがゼロである場合、逆量子化器はロケーションゼロで始まる適切なテーブルの連続するロケーションを読み出すであろう。各ロケーションは拡張トークンワードになる（9番目のビットはゼロになる）。このオペレーションの終わりで、トークンは正確に64の拡張トークンワードを含むであろう。

【1563】このトークンに答える逆量子化器のオペレーションは、ゼロと64を除く全ての拡張ワードナンバーのために限定されない。

【1564】B. 8. 7. 7 「JPEGテーブルセレクト」

このトークンは `iq jpeg indirection` への／からのテーブルナンバーに対して、色成分の翻訳をロードする／アンロードするために使用される。これらの翻訳はJPEG及び他のスタンダードにおいて使用される。

【1565】トークンヘッ드는現在関心のある色成分を特定する2つのビットを含む。これらは `iq component [1:0]` に置かれる。

【1566】トークンヘッ드의拡張ビットが1である場合、トークンは1つの拡張ワード、`iq jpeg indirection [2iq component [1:0] + 1:2iq component [1:0]]` ロケーションに書き込まれる最も低い2ビットを含む。たった今読み出されたばかりのバリューがトークン拡張ワードになる（上位7ビットがゼロになる）。このオペレーションの終了時に、トークンは正確に1つのトークン拡張ワードを含むであろう。

【1567】

【表222】

ヘッダー内の色成分	アクセスされる <code>iq jpeg indirection</code> のビット
0	[1:0]
1	[3:2]
2	[5:4]
3	[7:6]

表B. 8. 3 JPEG TABLE SELECT動作

B. 8. 7. 8 「MPEGテーブルセレクト」

このトークンはMPEGスタンダードを介してのプロセッシングの間に、デフォルトもしくはユーザーが限定する量子化テーブルを使用するか否かを定義するために使用される。トークンヘッ드는2ビットを含む。ヘッダーのビット0は `iq mpeg indirection` が書き込まれる場合どちらのビットに書き込まれるかを決定する。ビット1はそのロケーションに書き込まれる。

【1568】`iq mpeg indirection [1:0]` レジスタはシーケンススタートトークンによってクリアされるので、ユーザー限定量子化テーブルがビットストリーム内に送られた場合、このトークンを使用することだけが必要であろう。

【1569】B. 8. 8 「マイクロプロセッサレジスタ」

B. 8. 8. 1 「`iq access`」

いずれかの `iq` レジスタへのマイクロプロセッサアクセスを得るために、`iq access` は1に設定されなければならない。それが1を読み返すまで登録されなければならない（B. 8. 6. 2を参照）。これをするを怠ると、読まれているレジスタがデータベースによってまだ制御されている状態になり、安定しない。igramの場合、アクセスはロックアウトされ、ゼロを読み戻す。

【1570】`iq access` に0を書き込むと、制御をデータベースに手放すことになる。

【1571】B. 8. 8. 2 「`Iq coding standard [1:0]`」

このレジスタは逆量子化器によって実行されるコーディングスタンダードを保持する。

【1572】

【表223】

iq coding standard	コーディングスタンダード
0	H. 26.1
1	JPEG
2	MPEG
3	XXX

表B. 8. 4 コーディングスタンダードバリュウ

このレジスタはコーディングスタンダードバリュウによってロードされる。

【1573】これは2ビットレジスタではあるが、現在メモリーマップに8ビットが割り当てられており、将来の実装では上記スタンダード以上のものを処理することができるだろう。

【1574】B. 8. 8. 3 「Iq mpeg in direction [1:0]」

この2ビットレジスタはMPEGデコーディングオペレーションの間に、どの量子化テーブルが使用される予定であるかを記録するために使用される。

【1575】Iq mpeg indirection [0] はイントラコード化ブロックのために使用されるテーブルを制御する。それが0であれば、量子化テーブル0が使用され、デフォルト量子化テーブルを含むことが期待される。それが1であれば、量子化テーブル2が使用され、ユーザー限定量子化テーブルを含むことが期待される。

【1576】このレジスタはMPEG TABLE SELECTトークンによってロードされ、シーケンススタートトークンによって0に設定される。

【1577】B. 8. 8. 4 「Iq jpeg in direction [7:0]」

この8ビットレジスタはJPEGスキャンにおいて発生する4つの可能な色成分の各々のために、どの4つの量子化テーブルを使用するかを決定する。

【1578】・ビット [1:0] は成分0のために使用されるテーブルナンバーを保持する。

【1579】・ビット [3:2] は成分1のために使用されるテーブルナンバーを保持する。

【1580】・ビット [5:4] は成分2のために使用されるテーブルナンバーを保持する。

【1581】・ビット [7:6] は成分3のために使用されるテーブルナンバーを保持する。

【1582】このレジスタはJPEG TABLE SELECTトークンによって影響される。

【1583】B. 8. 8. 5 「Id quant scale [4:0]」

このレジスタは量子化スケール関数の現在のバリュウを

保持する。このレジスタはQUANT SCALEトークンによってロードされる。

【1584】B. 8. 8. 6 「iq component [1:0]」

このレジスタは量子化テーブルマトリックス (QTM) ナンバーに翻訳されるバリュウを保持する。それはトークンナンバーによってロードされる。

【1585】データトークンヘッダはこのレジスタに処理されようとしているブロックの色成分がロードされるようにする。この情報はQTMナンバーを決定するため、JPEG及びJPEGバリエーションにおいてのみ使用され、それはreference to iq jpeg indirection [7:0] で処理する。他のスタンダードでは、iq component [1:0] が無視される。JPEGテーブルセレクトトークンはこのレジスタに色成分をロードさせる。その後、それはトークンボディによってアクセスされる iq jpeg indirection [7:0] へのインデックスとして使用される。

【1586】量子化スケールトークンはこのレジスタにQTMナンバーをロードさせる。このテーブルはその後 (トークンの拡張フォームが使用される場合)、トークンからロードされるか、または適当に拡張されたトークンを形成するためにテーブルから読み出される。

【1587】B. 8. 8. 7 「iq prediction mode [1:0]」

この2ビットレジスタはそれに続くブロックのために使用される予測モードを保持する。逆量子化器がこの情報を利用する唯一の利用法は、イントラコーディングが使用されるか否かを決定することである。レジスタの両ビット共0であれば、次のブロックはイントラコード化されている。

【1588】このレジスタは予測モードトークンによってロードされる。このレジスタはシーケンススタートトークンによって0に設定される。

【1589】Id prediction mode [1:0] はJPEG及びJPEGバリエーションモードにおけるオペレーションに何の影響も与えない。

【1590】B. 8. 8. 8 「Id jpeg in

direction [7:0]」

Iq jpeg indirectionはQTMナンバーに色成分を翻訳するためのルックアップ表として使用される。従って、iq componentは、表222に示すように、iq jpeg indirectionに対するインデックスとして使用される。

【1591】このレジスタロケーションは、トークンの拡張フォームが使用される場合、JPEGテーブルセレクトトークンによって直接書き込まれる。

【1592】このレジスタロケーションは、トークンの非拡張フォームが使用される場合、JPEGテーブルセレクトトークンによって直接読み出される。

【1593】B. 8. 8. 9 「Iq quant table [3:0] [63:0] [7:0]」
4つの量子化テーブルがあり、各々が64のロケーションを持つ。各ロケーションは8ビットバリューである。バリュー0は如何なるロケーションにおいても使用されるべきではない。

【1594】これらのレジスタはB. 8. 6. 3のIgramにおいて説明したRAMとして実装される。

【1595】これらのテーブルは量子化テーブルトークンを用いてロードできる。

【1596】これらのテーブル内のデータはジグザグスキャンオーダーで記憶されることに注意。多くの文書は方形の8×8のナンバーアレイとして量子化テーブルバリューを表示している。通常、DCタームは左上位にあり、水平の周波数が左から右へと増加し、垂直の周波数が上から下へと増加する。該かるテーブルは、ナンバーが連続するiを持つ量子化テーブルの中に置かれるにつれて、ジグザグスキャンパスに沿って読まれなければならない。

【1597】B. 8. 9 「マイクロプロセッサレジスタマップ」

【1598】

【表224】

レジスタ	ロケーション	方向	リセット状態
iq access	0x30	R/W	0
iq coding standard [1:0]	0x31	R/W	0
iq quant scale [4:0]	0x32	R/W	?
iq component [1:0]	0x33	R/W	?
iq prediction mode [1:0]	0x34	R/W	0
iq jpeg indirection [7:0]	0x35	R/W	?
iq mpeg indirection [1:0]	0x36	R/W	0
iq qtm keyhole addr [7:0]	0x38	R/W	0
iq qtm keyhole [7:0]	0x39	R/W	?

表B. 8. 5 メモリーマップ

B. 8. 10 「試験」

入力における逆量子化器に対するテストカバリッジは逆量子化器の出力スノーパを通してであり、出力においては逆量子化器自体のスノーパを通してである。ロジックは逆量子化器自体のスキャンチェーンによってカバーされる。

【1599】Iamtest信号が認定されると、iq accessに関係なくIgramに対するアクセスが得られる。

セクションB. 9 「IDCT」

B. 9. 1 「序文」

逆離散コサイン変換 (IDCT) ブロックをここで説明する目的は、IDCT用のエンジニアリング情報源を提供することである。それは以下の情報を含む。

・IDCTの目的及び主な特徴

・それがどのように設計され、実証されたか

・構造

更に、その目的はそうした説明が以下の作業を容易にす

る、あるいは助けるために当業者に十分な情報を提供することである。

【1600】・「シリコンマクロ関数」としてのIDCTの認識

- ・別の装置へのIDCTの統合
- ・IDCTシリコン用のテストプログラムの開発
- ・IDCTの修正、再設計もしくは維持
- ・順方向DCTブロックの開発

B. 9. 2 「展望」

離散コサイン変換／ジグザグ (DCT/ZZ) はピクセルのブロック上で変換を行い、各ブロックは高さ8ピクセル×幅8ピクセルのスクリーンエリアを表示する。変換の目的は、周波数によって分類される周波数領域においてピクセルブロックを表示することである。目はピクチャ内のDC成分に敏感であるが、高周波成分に対してはあまり敏感ではないので、周波数データは目の感度に従って、各成分が別個にマグニチュードを減少させるようにする。マグニチュード減少プロセスは量子化として知られている。量子化プロセスはピクチャに含まれる情報を減少させる、つまり、量子化プロセスは損失性である。損失性プロセスはある種の情報を削除することにより全体的なデータ圧縮を行う。周波数データは高周波が0に量子化されやすくするように分類され、全てが連続して現れる。連続する0は、ランレングスコーディングは一般的に損失性のプロセスではないが、ランレングスコーディング計画を使用することにより量子化されるデータをコーディングすることが、更なるデータ圧縮を生じさせることを意味する。

【1601】IDCTブロック (これは実際に逆ジグザグRAMまたはIZZ、及びIDCTを含む) が分類される周波数データを取り、それを空間的データに変換する。この逆分類プロセスはIZZの機能である。

【1602】IDCTブロックがその一部を形成するピクチャ減圧システムは、ピクセルを整数として指定する。これはIDCTブロックが整数値を取り、また整数値を生じなければならないことを意味する。しかしながら、IDCTの関数は整数に基づいていないので、内部のナンバー表示は内部精度を維持するため分数部分を使用する。完全な浮動小数点計算が好ましいが、ここに記載する実装では固定小数点計算を使用する。固定小数点計算を使用すると少しばかり精度の損失があるが、この実装に関する精度はH. 261及びIEEEによって指定される精度を越えている。

【1603】B. 9. 3 「デザイン目標」

本発明によれば、主なデザイン目標は最小のシリコンエリアを使用する機能的に正確なIDCTブロックを設計することであった。更に、デザインは指定された操作条件の下で30MHzのクロック速度でランすることが求められていたが、更に将来に対する適用性も持つべきであると考えられた。より高いクロック率が将来必要とな

るであろうし、デザインの構造は可能な所ではこれを可能にするものである。

【1604】B. 9. 4 「IDCTインターフェースの説明」

IDCTブロックは以下のインターフェースを持っている。

【1605】・12ビット幅のトークンデータ入力ポート

- ・ビット幅のトークンデータ出力ポート
- ・マイクロプロセッサインターフェースポート
- ・システムサービス入力ポート
- ・テストインターフェース
- ・再同期信号

両トークンデータポートは前述した標準の2線式インターフェース型である。図示した幅はポートの全ワイヤ数ではなく、データ表示におけるビット数を指している。それに加えて、入力トークンデータポートに連合するのは、前のブロックの出力に対する再同期のために使用されるクロック信号及びリセット信号である。更に、出力トークンデータポートに連合し、次に続くブロックによって使用される2つの再同期クロックがある。

【1606】マイクロプロセッサインターフェースは標準であり、4ビットのアドレスを使用する。更に、3つの外部的にデコードされるセレクト入力があり、それらはイベント、内部レジスタ、及びテストレジスタ用のアドレススペースを選択するために使用される。このメカニズムはIDCTアドレススペースを異なるチップ内の異なる位置にマップするためのフレキシビリティを提供する。更に、1つのイベント出力、idcteventと、2つのi/o信号、n derrdとn serrdがあり、それらはIDCT及びマイクロプロセッサの非データパスの適当なビットに外部的に接続されるイベントトライステートデータワイヤである。

【1607】システムのサービスポートは標準のクロック信号とリセット入力信号、及び2フェイズオーバーライドクロック及び関連するクロックオーバーライドモードセレクト入力で構成される。

【1608】テストインターフェースはJTAGクロック信号及びリセット信号、スキャンバスデータ、制御信号及びramtestとchiptest入力で構成される。

【1609】通常の実行では、IDCTがその指定された機能を果たすためにマイクロプロセッサのアクセスを必要としないので、マイクロプロセッサポートは不活性である。同様に、テストインターフェースはテストもしくは確認が必要な場合にのみ活性である。

【1610】B. 9. 5 「離散コサイン変換用の演算的基盤」

ビデオ帯域幅圧縮において、入力データはピクチャの方形部分を表示する。従って、適用される変換は二次元で

なければならない。二次元変換は効果的に計算するのは困難であるが、二次元DCTは分離できるプロパティを持っている。分離可能な変換は他の次元とは別に各次元に沿って計算され得る。これを実行するには、ハードウェアにマッピングするために特に設計される一次元のIDCTアルゴリズムを使用し；そのアルゴリズムはソフ

EQ 10. forward DCT

$$Y(j, k) = \frac{2}{N} c(j) c(k) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} X(m, n) \cos \left[\frac{(2m+1)j\pi}{2N} \right] \cos \left[\frac{(2n+1)k\pi}{2N} \right]$$

EQ 11. inverse DCT

$$X(m, n) = \frac{2}{N} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} c(j) c(k) Y(j, k) \cos \left[\frac{(2m+1)j\pi}{2N} \right] \cos \left[\frac{(2n+1)k\pi}{2N} \right]$$

$$j, k = 0, 1, \dots, N-1$$

$$c(j) c(k) = \begin{cases} \frac{1}{\sqrt{2}} & j, k = 0 \\ 1 & \text{otherwise} \end{cases}$$

上記定義は演算的に、かけ算の間にマトリックス互換を行い、2つのN×Nマトリックスを連続して二度掛けるのに等しい。一次元DCTは演算的に2つのNNマトリックスを掛けるのに等しい。演算的に、二次元の場合は：

$$Y = [XC] TC$$

式中、Cはコサインの項のマトリックスである。

【1613】このように、DCTは時にはマトリックス操作の項で説明されることがある。マトリックスの説明は変換の演算的約分のために便利であるが、これは記法を簡単にするだけであることを強調しておかなければならない。2/Nの項がDCレベルを支配することに注意。定数c(j)及びc(k)は正規化関数として知られている。

【1614】B. 9. 6 「IDCT変換アルゴリズム」

下記に続けて詳細に説明するように、実際のIDCT変換を計算するために使用されるアルゴリズムは「高速」アルゴリズムであるべきである。使用されるアルゴリズムは効率的なハードウェア構造及び実装のために最適化される。アルゴリズムの主な特徴は、1つの乗法を取り除くための=2スケーリングの使用、及び上位及び下位セクションの間により大きな対称を生じさせるために設計されるアルゴリズムの変換である。この対称は最も高価な演算部の多くを効率的に再使用できるようにする。

【1615】アルゴリズムを示す線図(図145)において、上位半分と下位半分の間に対称は中間部分で明らかである。アダーとサブトラクターの最後のコラムも対

トウェアモデルには適切ではない。一次元アルゴリズムは二次元の結果を得るために連続的に適用される。

【1611】N×Nのピクセルブロック用の二次元DCTの演算的定義は以下の通りである：

【1612】

【数5】

称部分を有しており、アダーとサブトラクターは比較的低価格で組み合わせることができる(4つのアダー/サブトラクターは図示されるような4つのアダーと4つのサブトラクターよりかなり小さくなる)。

【1616】一次元変換の全ての出力は=2によって概算されることに注目。これは最後の二次元の答えが2によって概算されることを意味する。これは最終的なサチュレーション及びシフティングによるラウンディングステージにおいて容易に訂正できる。

【1617】図示されたアルゴリズムは二倍精度浮動小数点Cにおいてコード化され、この結果は(明瞭なマトリックス乗法を用いて)基準IDCTと比較された。次のステージはC(タイミング情報は含まれない)におけるアルゴリズムのビット正確整数バージョンをコード化するために使用され、それはアルゴリズムがシリコン上に実装されるであろう時に、アルゴリズムの性能及び精度を確認するために使用できたであろう。許容できる変換の間違ひはH. 261スタンダードにおいて明記されており、この方法はビット正確モデルを実行し、伝えられる精度を測定するために使用される。

【1618】図146は上位セクションと下位セクション間の共通性を図示する方法で、全体的なIDCT構造を示し、更に中間結果を記憶することが必要なポイントを示している。回路は上位セクション及び下位セクションが別個に計算されるようにするため時分割多重化されている。

【1619】B. 9. 7 「IDCT変換構造」

前述したように、IDCTアルゴリズムは効率的な構造

のために最適化される。その結果生じる構造の主な特徴は次の通りである：

- ・高価な演算操作の重要な再使用
- ・少数の乗算器、全てが多目的というよりむしろ一定係数である（乗算器サイズを減少させ、別の係数記憶の必要性を除去する）
- ・少数のラッチ、構造をパイプラインでつなぐためにわずかに必要である
- ・パイプラインステージ毎に1つの分解操作だけが必要であるようにオペレーションが配置される
- ・自然のオーダーで結果を生じるように配置できる
- ・複雑なクロスバースイッチングもしくは重要なマルチプレクシングがない（両者共、最終的な実装において高価である）
- ・2つのキャリセーブオペレーション（1つは加算、1つは減算）を取り除くために、分解結果から引き出される利点
- ・各ステージが4クロックサイクルを取る、つまり非常に高速の（大きな）演算操作の必要性を除去することを可能にする構造
- ・小さくて遅い波動的桁上げから、大きくて高速のキャリルックアヘッドバージョンへと単に分解操作を変更することにより、現行の30MHzピクセルクロックオペレーションよりはるかに高速のオペレーションを支持するであろう構造。分解操作は各ステージにおいて必要な時間の最大部分を必要とするので、これらのオペレーションだけをスピードアップすることは全体的なオペレーションスピードに重大な影響を及ぼす一方、変換の全体的なサイズの増加は比較的小さくてすむ。更に、速度の増加はパイプラインの深さを増加させることによって達成できる。

【1620】・変換データフローの制御は非常に明瞭であり、効率的である。

【1621】一次元変換微細構造の線図（図150）は、アルゴリズムが小さな一組のハードウェア資源にマップされ、必要な性能が得られる方法を示している。この構造の制御は「制御シフトレジスタ」をデータフローパイプラインに適合させることにより達成される。この制御はデザインに対して明瞭であり、シリコンレイアウトにおいて効率的である。

【1622】図150上で指名される制御信号（latch, sel byp等）は、ラッチを制御するために使用される様々なイネーブル信号であり、信号が流れる。ラッチに対するクロック信号は図示されていない。

【1623】変換構造が変換サイズを最小にしながら、必要な精度基準を満たせるようにするという点から、幾つかの実装に関する詳細が重要となる。一般に使用される技術は主として2つのクラスに分かれる。

【1624】・固定小数点位置を個々に制御することにより、各々の中間状態で固定ワード幅を持つ最大ダイナ

ミックレンジの保持。

【1625】・（変換全体のワード幅を単に増加させることによる精度増加ではなく）演算オペレーションの選択操作により精度を達成するために、精度要件の統計的定義の利用。

【1626】変換を設計する明瞭な方法は、精度を達成するために充分な大きさにされた固定ワード幅で単に固定小数点を実行することから成ったであろう。不運なことに、このアプローチは結果的により大きなワード幅を生じさせ、従って大きな変換を生じさせる。本発明において使用されるアプローチは、特別の中間値のために利用できるダイナミックレンジを最大限利用する方法で、変換を通じて固定小数点位置が変化できるようにし、最大限可能な精度を達成する。

【1627】利用できる結果が統計的に明記されるので、全体的な精度を改善するためにどの中間値に対しても選択的調整が行われ得る。選択される調整はLSB計算の単なる操作であり、それはほとんど費用のかからないものである。この技術の代替案としては、ワード幅を増やすことがあるが、かなりの費用がかかる。調整は最終的な結果が以前に所定の方角と反対の方角に行く傾向が見い出されていれば、効果的にその結果に対して所定の方角に加重値を与えることである。結果の断片的な部分を調整することにより、これらの結果の全体的な平均を効果的にシフトすることができる。

【1628】B. 9. 8 「IDCTブロック線図の説明」

IDCTのブロック線図はトークンストリームのプロセッシングに関連する全てのブロックを示す。この線図、図147はクロッキング、テスト及びマイクロプロセッサアクセス、及びイベントメカニズムを詳細には示していない。テストアクセスを提供するために使用されるスノーバブロックは線図において図示されていない。

【1629】B. 9. 8. 1 「データ誤差チェッカー」

最初のブロックはデータ誤差チェッカー及びコレクターであり、12ビット幅のトークンストリームを選び/作り出し、このストリームを分析し、データトークンをチェックするdecheckと呼ばれる。他の全てのトークンは無視され、直接送られる。遂行されるチェックは64に等しくない拡張数を持つデータトークンのために行われる。可能な誤差はdeficient（不足）

（<64拡張）、idct too few event、及びsupernumerary（過剰）（>64拡張）、idct too many eventと呼ばれる。該かる誤差は標準のイベントメカニズムで合図されるが、ブロックもトークンストリームの操作により簡単な誤差リカバリを試みる。不足誤差の場合、データトークンは「0」バリエーション拡張が詰められ（入力受け入れを停止し、補入を実行する）正確な64拡張を作り

上げる。過剰誤差の場合、拡張ビットは64番目の拡張に「0」が強制され、全ての余分な拡張はトークンストリームから取り除かれる。

【1630】B. 9. 8. 2 「逆ジグザグ」

空間デコーダの次のブロックは逆ジグザグRAM、izzであり、これも12ビット幅のトークンストリームを選び/作り出す。他の全てのブロックと同様、ストリームは分析されるが、データトークンだけが認識される。他の全てのトークンは変更されずに送られる。データトークンも送られるが、拡張オーダーが変更される。このブロックは正確なデータトークン（つまり、64拡張のみ）を頼りとする。これが真実のものでなければ、オペレーションは指定されない。再配置は標準の逆ジグザグパターンに従って行われ、IDCT出力において水平に走査されるデータを提供するためにデフォルトによって行われる。更に垂直に走査される出力を提供するためオーダーリングを変更することも可能である。標準のIZZオーダーリングに加えて、このブロックは各8ワードローの余分な再配置を実行する。これはIDCT一次元変換ブロックの特別な要件の故に行われ、(0、1、2、3、4、5、6、7)というオーダーではなく、(1、3、5、7、0、2、4、6)というオーダーで出力されるローを生じる。

【1631】B. 9. 8. 3 「入力フォーマッティング部」

次のブロックは入力フォーマッティング部、ipfmtであり、それはIDCT変換の第1次元のためにデータ入力を形成する。このブロックは12ビット幅のトークンストリーム入力と22ビット幅のトークンストリーム出力を持つ。データトークンはIDCT変換標準22ビット幅ワードにおける正しい有効へと整数部分を動かすために、左にシフトされ、分数部分は0に設定される。これはこのポイントにおいて10ビット分数があることを意味する。他の全てのトークンはシフトされず、余分な未使用のビットは単に0に設定される。

【1632】B. 9. 8. 4 「1次元変換 - 第1次元」

図示されている次のブロックは最初の1次元IDCT変換ブロック、onedである。これは22ビット幅のトークンストリームを入力/出力し、いつものように、ストリームは分析され、データトークンが認識される。他の全てのトークンは無変更のまま送られる。データトークンは、8x8の逆離散コサイン変換の1次元実行を果たす、パイプラインでつながれたデータバスを通過する。第1次元の出力には、データワード内に7ビット分数がある。他の全てのトークンは単にデータ変換待ち時間に適合し、出力前にトークンストリームに組み換えられる単なるシフトレジスタデータバスを通過する。

【1633】B. 9. 8. 5 「転置RAM」

転置RAM tthmはそれがトークンストリームを処理

する方法において、逆ジグザグRAMと多くの面で似ている。処理されるトークンの幅(22ビット)及び遂行される再配置は異なるが、他の面では同じように作用し、実際にその制御ロジックの多くを共有する。ここでも、ローはカラムのローへの基本的スワッピングと共に、次のIDCT次元の要件のために付加的に再配置される。

【1634】B. 9. 8. 6 「1次元変換-第2次元」

図示されている次のブロックは1次元IDCT変換ブロックの別の例であり、全ての面で第1次元と同じである。この次元の出力には、4ビット分数がある。

B. 9. 8. 7 「ラウンドとサチュレーション」

ラウンド・サチュレーションブロック、rasは22ビット固定小数点フォーマットのデータ拡張を含む22ビット幅のトークンストリームを選び、9ビット幅のトークンストリームを出力するが、その場合データ拡張は(+ve無限大に向かって)整数にラウンドされ、9ビットの2の補数表示に飽和されており、他の全てのトークンは直接送られる。

【1635】B. 9. 9 「ブロックのハードウェア解説」

B. 9. 9. 1 「標準のブロック構造」

トークンストリームを処理する全てのブロックのために、図148に示すような標準の概念的な構造がある。これはトークンストリームの操作を遂行するセクションから2線式インターフェースラッチを分離する。この構造上のバリエーションには、余分な内部ブロック(RAMコア等)が含まれる。図示した幾つかのブロックでは、全ての「データバス」ロジックと一緒にグルーピングするという要件のため、その構造は(今も実際に存在するにもかかわらず)概略図においてあまり明瞭にはされず、これを全ての標準のセルロジックから分離する。ras等の非常に簡単なブロックの場合、論理的操作をせずに、ラッチされたoutacceptを直接入力2線式ラッチに置くことができる。

【1636】B. 9. 9. 2 「Decheck-データ誤差チェック/リカバリ」

トークンストリーム内の最初のブロックは、ブロック線図展望セクションで明記したように、データチェックと訂正を行う。検出される誤差は標準のイベントメカニズムで処理され、それはイベントにマスクすることができ、ブロックは誤差が検出される時、リカバリ手順で続けるか、あるいはイベントマスクステータス次第で停止することができる。IDCTは正しくないデータトークンを見るべきではないので、それが試みるリカバリは重大な問題であるかもしれないものを含もうとするかなり簡単な試みにすぎない。

【1637】このブロックは2ステージのパイプラインの深さがあり、zcelisにおいて完全に実行され

る。入力 2 線式インターフェースラッチは「フロント」タイプのものであり、(IDCT の前にある) このブロックがその前にあるものとは別の電源組織の上にある時、全ての入力がトランジスタゲートに到達して安全なオペレーションができるようにすることを意味している。このブロックはトークンストリームを分析し、非データトークンを直接送ることによって作用する。データトークンが発見されると、カウンタはヘッダの後見つけられた拡張数からスタートする。カウンタが 63 ではない時、拡張ビットが「0」であると見い出されると、誤差信号が発せられ (それはイベントロジックに行き)、そのイベント用のマスクビットの状態次第で、`decheck` が停止される (つまり、もはや入力をアクセプトしないか、出力を発しない) か、あるいは誤差リカバリを開始する。`deficient` 誤差用のリカバリメカニズムは、正しい拡張数のトークンストリームへの挿入を制御するためにカウンタを使用する (挿入されるバリューは常に「0」である)。明らかに、入力はアクセプトされないが、一方でこの挿入は続けられる。64 番目の拡張の拡張ビットが「0」ではないと見い出されると、`supernumerary` 誤差が発せられ、データトークンは拡張ビットを「0」に強制することによって完了され、「1」に設定された拡張ビットを持つ全ての後に続くワードは、データをアクセプトし続けるが出力を無効にすることによって、トークンストリームから削除される。

【1638】2つの誤差信号は (ブロックが停止されない限り) 永続的ではなく、つまり誤差信号だけが、リカバリが完了するまで、誤差が発出されるポイントから活性のまま残る。これは最低限度の 1 つの完全なサイクルであり、無限大に過剰なデータトークンの場合に永久に存続する。

【1639】B. 9. 9. 3 「`lzz` と `tram` - RAM の再配置」

`lzz` (逆ジグザグ RAM) 及び `tram` (転置 RAM) は、同じ機能のバリエーションを遂行し、違いより類似性の方を多く持っているので、ここでは一緒に考慮する。これらのブロックはトークンストリームを選び、他の全てのトークンを無変更のまま送る一方で、データトークンの拡張を再配置する。処理される拡張幅及び再配置のシーケンスは異なるが、各 RAM 用の制御ロジックの大きなセクションは同じであり、実際に各 RAM 用の概略図において例証される「コモン制御」ブロックに組織化される。幅の違いはこの制御セクションには何の影響も及ぼさないで、RAM コア及び適当な幅の 2 線式インターフェースブロックと共に、各 RAM 用に異なる「シーケンスアドレス発生器」を使用することだけが必要である。

【1640】各 RAM の全体的な行為は本質的に FIFO のものと同じである。このことはトークンレベルにお

いて本当であり、出力オーダーに対する特別な修正はデータトークンの拡張ワードのために行われる。FIFO の深さは 128 ステージである。これはデータトークンの出力スタートが検出された後 FIFO の出力が支持されるので、システムを通じて持続できる 30 MHz のための要件を満たすことが必要である。これは、使用される再配置シーケンスの特徴が、再配置出力を始めることができる前に、FIFO に完全な 64 拡張ブロックが集められることを要求しているからである。より正確に言えば、逆ジグザグと転置シーケンス用の必要な最低限の数が異なっており、両者の場合 64 より幾分少ない。しかしながら、2 のべきではない長さを持つ FIFO を制御するという複雑性は、RAM コアにおける小さな節約が、必要な制御ロジックの付加的な複雑さより重要であろうということを意味する。RAM コアは 1 つの 30 MHz サイクルにおいて (同じアドレスまたは別のアドレスへの) リード及びライトができるようにするデザインで実行される。これは RAM が内部 60 MHz のサイクル時間で効果的に操作することを意味する。再配置オペレーションは 0 ~ 63 の範囲で、しかし自然のオーダーではなく、リードアドレスの特別なシーケンスを発生させることにより (`sequence address generation`) 遂行される。必要なシーケンスは (8 回の水平または垂直スキャンのための) 標準ジグザグシーケンスによって、あるいは通常のマトリックス転置のために必要なシーケンスによって指定される。これらの標準シーケンスは、IDCT 変換 1 次元ブロックの要件の故に、奇数/偶数フォーマット内で各ローを出力するための要件 (つまり、(0, 1, 2, 3, 4, 5, 6, 7) ではなく (1, 3, 5, 7, 0, 2, 4, 6)) によってその後更に再配置される。

【1641】転置アドレスシーケンス発生はアルゴリズム的に全く明瞭である。直接的な転置シーケンス発生は単にロー及びカラムアドレスの別々の発生を必要とするだけであり、それらは共にカウンタで実行される。ロー再配置の要件は単に、ローアドレスが自然のカウンタではなく、むしろ簡単な特殊なステートマシーンで発生されるということの意味する。

【1642】逆ジグザグシーケンスはアルゴリズム的に発生するため、あまり明瞭ではない。この事実の故に、アドレスの全 64 ビットバリューを保持するために小さな ROM が使用され、これは水平と垂直のスキャンモード間で変化するために、スワップされ得るローとカラムのカウンタでアドレスされる。ROM ベースの発生器は非常に素早く設計され、順方向ジグザグ (ROM 再プログラム) を実行するか、もしくは他の代替的なシーケンスを将来加えることは些細なことであるという利点をも有する。

【1643】B. 9. 9. 4 「`OneD` - 1 次元 `lbc` 変換」

このブロックは20ステージのパイプライン深さを持ち、パイプラインは失速されると硬くなる。この剛性はデザインを大きく簡略化し、パイプラインの深さはそれほど大きくないので、全体的な力に過度に影響を及ぼすべきではなく、両次元は一定量のバッファリングを提供するRAMの後を引き継ぐ。

【1644】ブロックは標準構造に従うが、(プロセスされるべき)データトークン拡張及び無変更のまま送られるべき他の全てのアイテムのために、内部的に別のパスを有する。概略図は特別な方法で描かれていることに注意。まず、全てのデータパスロジックを一緒にグループ分けするという要件の故に、次に、自動的に編集されたコード発生を可能にするという要件(これはトップレベルで制御ロジックを説明する)の故に。

【1645】トークンはいつもの通り分析され、それからデータ拡張とその他のバリューが、出力2線式インターフェースラッチブロックの前のマルチプレクサで組換えられる前に、2つの異なる並列バスを通して各々送られる。変換データバスを通してバリューを無変更のまま送ることはできないので、並列バスが必要である。変換データバスの待ち時間は、トークンストリームの残りを処理するために、簡単なシフトレジスタと適合される。

【1646】onedの制御セクションはトークンストリームを分析し、トークンのスプリッティングと組換えを制御する必要がある。その他の主なセクションは変換データバスを制御する。このデータバスの制御用の主メカニズムはデータバスパイプラインを適合させる制御シフトレジスタであり、データバスパイプラインの各ステージ用に必要な制御信号を提供するためにタップが外される。

【1647】onedブロックはデータ拡張の完全なロー、つまり8のグループに関するオペレーションをスタートすることだけができるという要件を持つ。ローの中間で無効なデータ(Gaps)を処理することはできないが、事実上、izz及びtramのオペレーションは完全なデータブロックが64の有効な拡張バリューの割り込みなしのシーケンスとして出力されることを保証する。

【1648】B. 9. 9. 4. 1 「変換データバス」変換データバスの微細構造、tdpは図150において以前に図示した。一部の詳細(例えばクロッキング、シフト等)は図示していないことに注意。しかしながら、この線図はパイプラインのどのステージにおいても、データバスが4つのバリューに関して同時に操作する方法を図示している。基本的なデータバスのサブ構造、つまり、3つの主要セクション(例えば、プリコモン、コモン、ポストコモン)も、演算及びラッチリソースが必要になるにつれて、見ることができる。指名された制御信号は制御シフトレジスタステートのデコードで順序付けされるパイプラインラッチ(及び加算/減算セ

レクタ)用のイネーブルである。各パイプラインステージは長さにおいて実際に4つのクロックサイクルであることに注意。

【1649】変換データバス内に、入力を集め、パイプラインに中間結果を記憶し、出力を順番に並べることが必要な多くのラッチステージがある。幾つかのラッチはマックスタイプ、つまりそれらは条件付きで1つ以上のソースからロードされ得る。全てのラッチは可能化タイプのものである、つまり、別々のクロックとイネーブル入力がある。これは作られたクロックスキームが適用された場合に生じるであろう不均斉という問題を考慮しなければならないというより、正しいタイミングでイネーブル信号を発生させることの方がたやすいことを意味する。

【1650】必要とされる主要な演算要素は次の通りである。

【1651】・多数の固定化係数乗算器(キャリセーブ出力)

- ・キャリセーブアダー
- ・キャリセーブサブトラクター
- ・分解アダー
- ・分解アダー/サブトラクター

全ての演算は2の補数表示で行われる。これは通常の(分解された)形態もしくはキャリセーブの形態(つまり、その合計が実際のバリューを表示する2つの数字)であってよい。全ての数字は記憶の前に分解され、これは時間の点から最も高価なオペレーションであるので、1つの分解オペレーションだけがパイプラインステージ毎に行われる。分解オペレーションはここで行われ、全てのオペレーションが簡単な波及的桁上げを使用する。これは分解器が非常に小さいが、比較的遅いことを意味する。分解は各ステージにおいて全体の時間を支配するので、高速分解演算装置を使用して全体の変換をスピードアップする機会が明らかにある。

【1652】B. 9. 9. 5 「"Ras"-ラウンディングとサチュレーション」

本発明では、rasブロックは第2次元onedの出力から22ビットの固定小数点ナンバーを選択し、これらを正確にラウンドされ飽和された必要な9ビットのサイン済み整数結果に変えるという仕事を果たす。このブロックは更に、スキーム(2/Nターム)内の固有の必要な4で割る作業を遂行し、更に2次元の各々において遂行される=2のプリスケールリングを補うために必要な2で割る作業を遂行する。この8で割る作業は固定小数点位置が予測された以上に残された3ビットであると解釈される、つまりその結果を15ビットの整数表示と(4ビット分数ではなくむしろ)7ビット分数として処理することを含蓄している。実行されるラウンディングモードは「正の無限大にラウンドする」、つまり、正確に0.5の分数に1を加算することである。これは実行す

るための最も簡単なラウンディングモードであるので、基本的に行われる。ラウンディング（整数部分の条件付き増分）が完了した後、9ビットのサイン済み結果がこの範囲内において最大または最小バリューに飽和される必要があるかどうかを見るために、この結果が検査される。これは元の整数バリューの上位ビットと共に実施される増分の検査によって行われる。

【1653】いつものように、トークンストリームは分析され、ラウンドとサチュレーションオペレーションがデータトークン拡張値に対してのみ適用される。ブロックは深さが2ステージのパイプラインを持ち、完全に2

cellsで実行される。

【1654】B. 9. 9. 6 「Idctsel s - IDCTレジスタセレクトデコーダ」

このブロックは4つのマイクロプロセッサインターフェースアドレスライン及びsel test入力を、個々のブロックテストアクセス用のセレクトライン（スノーパ及びRAM）にデコードする簡単なデコーダである。ブロックは2 cells結合ロジックのみで構成される。デコードされるセレクトは表227に示す。

【1655】

【表225】

アドレス	ビットナンバー	レジスタ名
0x0	7..0	使用されない
	0	TRAMキーホールアドレス
0x1	7..0	
0x2	7..0	TRAMキーホールデータ
0x3	7..0	TRAMキーホールデータ
0x4	7..0	IZZキーホールアドレス
0x5	7..0	IZZキーホールデータ
0x6	7..3	使用されない
	2	ipfsnoopテストセレクト
	1	ipfsnoopバリッド
	0	ipfsnoopアクセプト
0x7	7..5	使用されない
	5..0	ipfsnoopビット [21:16]
0x8	7..0	ipfsnoopビット [15:8]
0x9	7..0	ipfsnoopビット [7:0]

注a. 繰り返されるアドレス

表B. 9. 1 IDCTテストアドレススペース (1/2)

【1656】

40 【表226】

アドレス	ビットナンバー	レジスタ名
0xA	7..3	使用されない
	2	d2snoop テストセレクト
	1	d2snoop バリッド
	0	d2snoop アクセプト
0xB	7..6	使用されない
	5..0	d2snoop ビット [21:16]
0xC	7..0	d2snoop ビット [15:8]
0xD	7..0	d2snoop ビット [7:0]
0xE	7	outsnoop テストセレクト
	6	outsnoop バリッド
	5	outsnoop アクセプト
	4..2	使用されない
	1..0	outsnoop データ [9:8]
0xF	7..0	outsnoop データ [7:0]

表B.9.1 IDCTテストアドレススペース (2/2)

B.9.9.7 「Idctregs-IDCT制御レジスタとイベント」

発明のこのブロックはデータの不足誤差及び過剰誤差、及びIDCT出力が垂直に走査されるように、izz再配置変更を行うために使用できる単一メモリーマップ済みビットvscanを処理する標準イベントロジックブロックの例を含む。このビットはバリュー「0」にリセットされる、つまりデフォルトモードが水平にスキャンアウトされる。2つの可能なイベントは割り込みとして使用できるidctevent信号を形成するために、共にORされる。レジスタ及びイベントのアドレス及びビット位置に関しては、セクションB.9.10を参照せよ。

【1657】B.9.9.8 「クロック発生器」
IDCTにおいて2つの「標準」タイプ(clkgen)のクロック発生器が使用される。これは2つの別々のスキャンバスがあるように実施される。クロック発生器はidctcga及びidctcgbと称される。機能的に、唯一の違いはidctcgbがnotrstl信号を発生させる必要がないことである。2つのクロック発生器におけるクロックとリセット出力の各々のためのバッファリング量は、各々のクロックもしくはリセットによって駆動される実際のロードに適合するように個

々に調整される。適合されるロードは最終レイアウトのゲート及びトラックキャパシタンスから実際に測定された。

【1658】IDCTトップレベルのBlock Place and Route (BPR) が実行された時の利点は、これらのトラックは有効電流を運ぶので、より重く負荷されたクロック(ph0bとph1b)用のクロック分配ツリーの最初のセクションのトラック幅を増加させるために、対話式大域ルーティング特徴の能力から引き出された。

【1659】B.9.9.9 「JTAG制御ブロック」

40 IDCTが2つの別個のスキャンチェーンと2つのクロック発生器を持つので、標準JTAG制御ブロック、jspectleには2つの場合がある。これらはテストポートと2つのスキャンバス間をつなぎ合わせる。

【1660】B.9.10 「イベント及び制御レジスタ」

IDCTは2つのイベントを発生させることができ、1つの制御ビットを持つ。2つのイベントとは、不正確なデータトークンが検出された場合に、IDCTの前のdecheckブロックによって発生されるidct_toboverfloweventとidct_tobmany

eventである。1つの制御ビットは垂直に走査される出力でIDCTを操作することが必要な場合に設定されるvscanである。従って、このビットはizzブロックを制御する。全てのイベントロジック及びメモリーマップ制御ビットはブロックidctregsの中に置かれる。

【1661】IDCTの観点から、これらのレジスタは

アドレス (hex)	ビットナンバー	レジスタ名
0x0	7..1	使用された
	0	vscan

表B. 9. 2 IDCT制御レジスタアドレススペース

【1663】

【表228】

アドレス (hex)	ビット名	レジスタ名
0x0	n derrd	idct. too few event
	n serrd	idct too many event
0x1	n derrd	idct too few mask
	n serrd	idct too many mask

表B. 9. 3 IDCTイベントアドレススペース

B. 9. 11 「実行」

B. 9. 11. 1 「ロジックデザインアプローチ」

全てのIDCTブロックのデザインにおいて、発明によれば、統一された簡単なロジックデザイン戦略があり、それは素早く明瞭な方法で「安全な」設計を行うことが可能であるということの意味したであろう。多数の制御ロジックのために、マスター・スレーブだけを用いる簡単なスキームが採用された。非同期的なセット/リセット入力だけが正しいシステムリセットに接続された。同じ機能をより効率的に遂行するために利口な非標準型回路構成を提供することも可能であったかもしれないが、このスキームは次のような利点を持っている。

【1664】・概念的にシンプルである

- ・設計しやすい
- ・操作速度はかなり明らかであり（ラッチ→ロジック→ラッチ→ロジックスタイルのデザイン参照）、自動分析に従う
- ・グリッチは問題ではない（SRラッチを参照）
- ・初期設定のためにシステムリセットだけを使用する
- ・スキャンパスが正しく作用できるようにする
- ・自動的に従うCコード発生を可能にする

透明なd-タイプラッチが使用された多数の場所があり、これらは下記に記した。

次のロケーションに置かれる。トライステートi/oワイヤはn derrdであり、これらのロケーションを適宜にリード及びライトするためにn serrdが使用される。

【1662】

【表227】

【1665】B. 9. 11. 1. 1 「2線式インターフェースラッチ」

スタンダードブロック構造は入力及び出力の2線式インターフェースのためにラッチを使用する。出力2線式ラッチと次に続く入力2線式ラッチの間にはロジックは存在しない。

【1666】B. 9. 11. 1. 2 「ROMインターフェース」

ROM回路のタイミング要件の故に、ラッチはROMの出力におけるIZZシーケンス発生器において使用される。

【1667】B. 9. 11. 1. 3 「変換データバス及び制御シフトレジスタ」

完全なマスター・スレーブ装置としてあらゆるパイプライン記憶ステージを実装することが可能であるが、必要とされる記憶量の故に、ラッチを使用することにより得られる重大な節約がある。しかしながら、このスキームはユーザーが幾つかの要素を考慮することを求めている。

【1668】・制御シフトレジスタはイネーブルとして使用されるため、両フェイズの制御信号を作り出さなければならない（つまり、このシフトレジスタにおいてラッチを使用する必要）

・タイミング分析はラッチの使用により複雑になる
 ・1つのラッチが同じフェイズの別のラッチに出力するので、`tpostc`はもはや自動的に編集済みコードを作り出さないであろう（イネーブルのタイミングの故に、これは回路の問題ではない）
 にもかかわらず、ラッチの使用により節約されるエリアは本発明においてこれらの要素を受け入れることを価値のあるものになっている。

【1669】B. 9. 11. 1. 4 「マイクロプロセッサインターフェース」

本インターフェースの性質のため、イベント及びレジスタブロック`idctregs`及びRAMコア用のキーホールロジックにおいて、ラッチ（及びリシンクロナイザー）のための要件がある。

【1670】B. 9. 11. 1. 5 「JTAGテスト制御」

これらのスタンダードブロックはラッチを利用する。

【1671】B. 9. 11. 2 「回路デザインの問題」

IDCTデザイン（スタンダードセル、データバスライブラリ、RAM、ROM等）において使用されたライブラリセルのデザインにおいて為された仕事は別にして、IDCTにおいてトランジスタレベル回路デザインのための要件はない。（`Hspice`を用いる）回路シミュレーションは変換データバスにおいて幾つかの公知のクリティカルパスの中から実行され、`Hspice`は更に許容される最大長に近いパスの場合に、クリティカルパスアナリシス（CPA）ツールの結果を確認するためにも使用された。

【1672】IDCTは通常のオペレーションにおいて完全に静的である（つまり、我々はシステムクロックを無期限に停止することができる）が、テストクロックが停止される（または非常に低速にされる）時に衰えるであろう走査可能なラッチに動的なノードがあることに注意。`Vt`降下（例えば、マックス出力）を呈する幾つかのノードの非再生性のために、IDCTは静的な場合に「マイクロ・パワー」ではないであろう。

【1673】B. 9. 11. 3 「レイアウトアプローチ」

本発明のレイアウト実行に対する全体的アプローチは、多くの`zcell`と少数のマクロブロックで構成された完全なIDCTをレイアウトするために、BPR（幾つかのマニュアル干渉）を使用することであった。これらのマクロブロックは手動編集されたレイアウト（例えば、RAM、ROM、クロック発生器、データバス）であったか、もしくは`oned`ブロックの場合、更なる`zcell`及びデータバスからBPRを使用して構築されていた。

【1674】データバスは`kdp11b`セルから構築された。それに加えて、`kdp11b`セルの局部的に限定

されたレイアウトバリエーションが定義され、これが価値のあるサイズベネフィットを提供すると認められた場合に使用された。各々の`oned`ブロック、`oned`において使用されるデータバスは、デザインの中のずばぬけて最大の1エレメントであり、このデータバスのサイズ（高さ）を最適化するためかなりの努力が払われた。

【1675】データバス内のエレメントの正確なオーダリングが相互連絡が処理される方法に影響を及ぼすので、変換データバスの組織、`tdp`はむしろ決定的である。最大許容値（理想的には8、非常に不便であるが10も可能である）があるので、最も密集したポイントで発生する`overs`（サブブロックに接続しない垂直ワイヤ）の数を最小にすることが重要である。データバスは3つの主要なサブセクションに論理的にスプリットされ、こうしてデータバスレイアウトが実行される。各サブセクションにおいて、現に4つの並列するデータフローがあり（それは様々なポイントで組み合わせられ）、従って、各サブセクション内でデータ・フロー（そして、全てのエレメントの位置）を組織する多くの方法がある。各サブセクション内のブロックのオーダリング、及び論理バスの物理的バスピッチへの割当は、正確に接続されるであろうレイアウトを達成することを可能にするために、レイアウトが始まる前に注意深く算出された。

【1676】B. 9. 12 「照合」

IDCTの照合はアルゴリズムのトップレベルの照合から最終レイアウトチェックまで、多くのレベルにおいて為された。

【1677】変換構造に関する初期作業はCにおいて行われ、完全精度及びビット正確整数モデルが開発された。`H. 261`精度規格に対する適合性を立証し、変換構造内の計算のダイナミックレンジを測定するため、ビット正確モデルに関して様々なテストが実施された。

【1678】多くの場合Mを書くことによって、デザインはサブブロックの行動科学的説明（例えば、データバス及びRAMの制御）を前進させた。該かる説明は、そのブロックの概略的説明のデザインに移動する前に、`Lsim`においてシミュレートされた。ある場合（例えば、RAM、クロック発生器）には、行動科学的説明はトップレベルのシミュレーションのためにも使用された。

【1679】ロジックシミュレーションを遂行するための戦略は、そのレベルで適切にシミュレートするであろう全てのものために概略図をシミュレートすることであった。ローレベルのライブラリセル（つまり、`zcell`及び`kdp11b`）は、この結果はるかに小さく素早いシミュレーションであるので、主としてその行動科学的説明を用いてシミュレートされた。それに加えて、行動科学的ライブラリセルはある回路の構造問題を

強調することができるタイミングチェックという特徴を提供する。信頼チェックとして、ライブラリセルのトランジスタ解説を用いて、あるシミュレーションが実施された。全てのロジックシミュレーションはゼミダイレイ方式で行われ、従って機能上の性能を照合することが意図されていた。リアルタイミングの行動の照合は他の技術を用いて行われた。

【1680】タイミング性能の部分的照合として、(RC Timingモードを用いて) Lsimスイッチレベルシミュレーションが行われたが、更に他の潜在的なトランジスタレベルの問題(例えば、グリッチ高感度回路)のためのチェックを提供する。

【1681】タイミング問題をチェックするための主な照合技術はCPAツール、datechk用のpathオプションの使用であった。これはより長い信号パスを識別する(あるものはすでに知られていた)ために使用され、ある重大な場合にCPA分析を照合するためにHspiceが使用された。

【1682】IDCT行為のバルクは装置を通じてトークンのフローにより訓練されているので、ほとんどのLsimシミュレーションは標準のソース→ブロック→シンク方法論で行われた。マイクロプロセッサインターフェース(構成、イベント及びテストロジック)を通してアクセスされる特徴、及びJTAG/スキャンを介してアクセスされるテスト特徴をテストするために、付加的なシミュレーションも必要である。

【1683】編集済みコードシミュレーションは、やはり標準のソース→ブロック→シンク方法及びLsim照合において使用された同じトークンストリームの多くを用いて、全IDCTのために当業者によって容易に達成され得る。

【1684】B. 9. 13 「テストニング及びテストサポート」

本セクションはテストニング用及び各々のブロックが如何にテストされるかの分析用に提供されるメカニズムを扱う。

【1685】テストアクセス用に提供される3つのメカニズムは次の通りである:

- ・RAMコアへのマイクロプロセッサアクセス
- ・スノーバブロックへのマイクロプロセッサアクセス
- ・制御及びデータバスロジックへのスキャンバスアクセス

IDCTには2つの「スノーバ」ブロックと1つの「スーパースノーバ」ブロックがある。図149はスノーバブロックの位置と他のマイクロプロセッサのテストアクセスを示している。

【1686】これらと2つのRAMブロックを用いて、トークンフローに関するそれらの行動をテストする目的で、各々の主要ブロックを隔離することが可能である。マイクロプロセッサアクセスを用いて、如何なるブロッ

クに対するトークン入力をも制御し、隔離されているそのブロックのトークンポート出力を観察することが可能である。更に、各ブロックの制御セクションにおける(ほとんど)全てのフリップフロップとラッチを通過し、またoned変換データバスパイプラインの場合にデータバスラッチの一部を通過する2つの別々のスキャンバスがある。2つのスキャンバスはaとbと表示され、前者はdecheckブロックからipfmtブロックへと動き、後者は最初onedブロックからrasブロックへと動く。

【1687】スノーバに対するアクセスは通常の方法で適切なメモリーマップされたロケーションにアクセスすることにより可能である。(適当なものとしてramtest入力を用いて)RAMコアの場合にも同じことが言える。スキャンバスは通常の方法でJTAGポートを通してアクセスされる。

【1688】様々なテスト問題に関連して各ブロックを論じる。

【1689】B. 9. 13. 1 「Decheck」

このブロックは入力と出力の2線式インターフェース用の2つのラッチがプロセッシングブロックを囲む標準構造(図148を参照)を持つ。いつものように、これらの2線式ラッチは可能化される時はいつでも単にデータを通り過ぎ、テストされるロジックの深さを持たない。如何なる走査も2線式ラッチに対して行われない。このブロックでは、「制御」セクションは全てがスキャンバスaの上にあるzcellの1ステージパイプラインから成る。制御セクション内のロジックは比較的シンプルであり、ほとんどの複雑なバスはおそらく6ビット増分器が使用されるデータ拡張カウンタの発生においてである。

【1690】B. 9. 13. 2 「Izz」

このブロックは標準構造の変形であり、2線式インターフェースラッチと制御セクションに加えられるRAMコアブロックを含む。制御セクションはアドレスシーケンス発生のために使用されるzcellと小さなROMで実装される。全てのzcellはスキャンバスa上にあり、ROMアドレスへのアクセス及びzcellラッチを介したデータがある。更に、ロジック、例えばナンバーの発生プラス増分または減少する能力のためのロジックがある。それに加えて、リードアドレス発生のために使用される7ビットのフルアダーがある。RAMコアはキーホールレジスタを通して、マイクロプロセッサインターフェースを介してアクセス可能である。表225、表226を参照。

【1691】B. 9. 13. 3 「lpfmt」

このブロックも標準構造を持っている。制御ロジックはかなり簡単なzcellロジック(全てがスキャンバスa上にある)で実装されるが、ここでのロジックは非常に長くシンプルであるので、データのラッチング及びシ

フティング/マッキングは、直接的なアクセスなしにデータバスにおいて行われる。

【1692】B. 9. 13. 4 「Oned」

ここでも、このブロックは標準構造に従い、ランダムロジックとデータバスセクションに分かれる。zcellロジックは比較的明瞭であり、全てのzcellがスキャンバスa上にある。変換パイプラインデータバス用の制御信号は、スキャンバス上にあるzcellラッチで構成される長いシフトレジスタから引き出される。それに加えて、幾つかのパイプラインラッチがスキャンバス上にあり、これはパイプラインの幾つかのステージ（例えば、乗算器とアダー）間のロジックがかなり深いものであることから行われる。非データトークンはシフトレジスタに沿って送られ、データバスとして実装され、これらのステージのいずれに対するテストアクセスも存在しない。

【1693】B. 9. 13. 5 「Tram」

このブロックはizzブロックと酷似している。しかしながら、この場合、アドレスシーケンスアドレス発生において使用されるROMはない。これはアルゴリズム的に実行される。全てのzcell制御ステートはデータバスb上にある。

【1694】B. 9. 13. 6 「Rras」

このブロックは標準構造に従い、完全にzcellで実行される。最も複雑な論理機能はラウンディングアップ時に使用される8ビット増分器である。他の全てのロジックはかなりシンプルである。全てのステートはスキャンバスb上にある。

【1695】B. 9. 13. 7 「他のトップレベルのブロック」

IDCTのトップレベルに現れる他の幾つかのブロックがある。スノーバはJTAG制御ブロックであるのと同様、明らかにテストアクセスロジックの一部でもある。更に、特別なテストアクセスを持たない2つのクロック発生器がある（但し、それらは様々なテスト特徴を支持している）。ブロックidctselはマイクロプロセッサアドレスをデコーディングするための組合わせzcellロジックであり、ブロックidctregsはマイクロプロセッサアクセス可能イベント、及びIDCT関連制御ビットを含む。

【1696】セクション B. 10 「序文」

B. 10. 1 「時間デコーダの展望」

本発明による時間デコーダの内部構造を図151に示す。

【1697】チップ・ブロック間の全てのデータフロー（及びブロック内の多くのデータフロー）は2線式インターフェースによって制御され（詳細については技術参考書及びセクションを参照）、図151内の各々の矢印は2線式インターフェースを表す。入ってくるトークンストリームは、外部システムクロックからのデータをブ

エイズロックループ（ph0/ph1）から引き出される内部クロックに同期させる入力インターフェースを通過する。トークンストリームは次にトップフォークを介して2つのバスにスプリットされる；1つのストリームはアドレス発生器に進み、他のストリームは256ワードFIFOに進む。FIFOはデータをバッファする一方、前のIまたはPフレームからのデータはDRAMから引き出され、予測アダー（P及びBフレーム）内の空間デコーダからの入力誤差データに加えられる前に、予測フィルタにおいてプロセスされる。MPEGデコーディングの間に、出力フレームが正しいオーダーにあるように、フレーム再配置データがI及びPフレームのために引き出されなければならない。再配置されたデータはリードラダーブロック内のストリームに挿入される。

【1698】アドレス発生器はフォワード及びバックワード予測、再配置、リード及びライトバックのために別々のアドレスを発生させ、ライトバックされるデータはライトラダーブロック内のストリームからスプリットされる。最後に、データは出力インターフェースブロック内の外部クロックに再同期化される。

【1699】時間デコーダ内の全ての主要ブロックは内部マイクロプロセッサインターフェース（UPI）バスに接続される。これはマイクロプロセッサインターフェースブロック内の外部マイクロプロセッサインターフェース（MPI）から引き出される。このブロックはそれに関連するチップ内の様々なブロックのためにアドレスデコードを持つ。更に、マイクロプロセッサインターフェースと連合するのはイベントロジックである。

【1700】時間デコーダの残りのロジックは基本的にテスト関連である。第1に、IEEE1149.1（JTAG）インターフェースがJTAGバウンダリスキャン特徴に対すると共に、内部スキャンバスに対するインターフェースを提供する。第2に、テストモードの間に、マイクロプロセッサインターフェースを介してデータフローに対する侵入的アクセスを可能にする2線式インターフェースステージが、パイプライン構造内の戦略ポイントに含まれる。

【1701】セクション B. 11 「クロッキング、テスト及び関連問題」

40 B. 11. 1 「クロック様式」

チップ内の個々の機能的ブロックを考慮する前に、チップ内のクロック様式及びそれらの関係を認識することが有益であろう。

【1702】通常のおペレーションの間に、チップのほとんどのブロックはフェイズロックループ（PLL）からの信号pllsysclkに同期して動く。これに対する例外はDRAMインターフェースであり、そのタイミングはlfttleサブブロックに同期する必要によって支配され、このサブブロックはDRAM制御信号（hbtwe、hbtbe、hbtcas、hbtfa

s) を発生させる。このブロックのコアは2フェイズ・ノンオーバーラッピングクロック $clk0$ と $clk1$ によりクロックされ、それらは PLL $ckio$ 、 $ckil$ 及び $ckg0$ 、 $ckg1$ から別個に供給されるクアドラチュア2フェイズクロックから引き出される。

【1703】 $clk0$ 、 $clk1$ DRAMインターフェースクロックが残りのチップ内のクロックに同期するので、DRAMインターフェースと残りのチップ間のインターフェースにおいて、(實際上可能な限り) 準安定行為の可能性を除去するための対応策が取られた。同期化は2つの領域で発生する：アドレス発生器の出力インターフェースにおいて ($addrgen/predread/psgsync$ 、 $addrgen/ipwrt2/sync18$ 及び $addrgen/iprd2/sync18$)、及びDRAMインターフェース内のスイング・バッファRAMの「スインギング」を制御するブロックにおいてである (DRAMインターフェースに関するセクション参照)。各々の場合に、同期化プロセスは3つの直列準安定ハードフリップフロップによって達成される。注意すべきことは、これは $clk0/clk1$ がアドレス発生器の出力ステージにおいて使用されることを意味することである。

【1704】 これらの完全に同期したクロック様式に加えて、 $pllsysclk$ から2フェイズ・ノンオーバーラッピングクロック ($ph0$ 、 $ph1$) を発生させる多くの別個のクロック発生器がある。アドレス発生器、予測フィルタ及びDRAMインターフェースは各々自身のクロック発生器を持っており；残りのチップはコモンクロック発生器から離れて動かされる。こうした理由には2つの部分がある。第1に、個々のクロック発生器にかかる容量性負荷を減少させ、より小さなクロックドライバと減少したクロックルーティング幅を可能にする。第2に、各々のスキャンバスはクロック発生器によって

制御され、従って、クロック発生器の数の増加により短いスキャンバスを使用できるようになる。

【1705】 これらのクロック様式バウンダリを横切って動かされる信号を再同期化することが必要である。なぜなら、異なるクロック発生器から引き出されるノンオーバーラッピングクロック間の重要でないスキューがインターフェースにおいてアンダーラップが発生したことを意味するであろうからである。各「スノーパ」ブロック (セクション B. 11. 4 を参照) に内蔵された回路が、こうしたことが発生しないことを保証し、スノーパブロックは、トークンデコードブロックにおいて再同期化が行われるアドレス発生器の前を除いて、全てのクロック様式間の境界に置かれた。

【1706】 B. 11. 2 「クロックの制御」
各標準クロック発生器は通常モード及びスキャンテストモードで、オペレーションができるようにする多くの異なるクロックを発生させる。スキャンテストモードでのクロックの制御は他のセクションで詳細に説明するが、注目すべきことは、クロック発生器 ($tph0$ 、 $tph1$ 、 $tckm$ 、 $tcks$) により発せられるクロックの幾つかは、通常概略図の原始記号に加えられない。これはこれらのクロックを正確に接続するポストプロセッサにより、スキャンバスが自動的に作られるからである。機能的な観点から見ると、ポストプロセッサが概略図に示されたものとは異なるクロックを接続したという事実は無視することができる；行為は同じである。

【1707】 通常オペレーション中に、マスタークロックは多くの異なる方法で引き出すことができる。表229はピン $plselect$ 及びオーバーライドのステートにより様々なモードを選択できる様子を示している。

【1708】

【表229】

pllselect	オーバーライド	モード
0	0	pll sysclk が直接外部 sysclk に接続される。PLL を迂回する；DRAM インターフェイスクロック (cki0, cki1, ckq0, ckq1) はピン ti と tq から直接制御される。
0	1	オーバーライドモード - ph0 及び ph1 クロックがピン tph0ish と tph1ish から直接制御される；DRAM インターフェイスクロック (cki0, cki1, ckq0, ckq1) はピン ti と tq から直接制御される。
1	0	通常のオペレーション。pll sysclk は PLL によって作られるクロックである；DRAM インターフェイスクロックは PLL によって作られる。
1	1	ti と tq に接続される外部レジスタは内部レジスタ (debug のみ) の代わりに使用される。

表B.11.1 クロック制御モード

B.11.3 「2線式インターフェース」

2線式インターフェースの全体的な機能性については技術参考書において詳細に説明する。しかしながら、2線式インターフェースは時間デコーダ内の全てのブロック間の通信のために使用され、ほとんどのブロックは多くのパイプラインステージで構成され、その全ては2線式インターフェースステージである。従って、多くの概略図を解釈できるためには、2線式インターフェースの内部実装を理解することが重要である。一般的に、これらの内部パイプラインステージは図152に示すように構成される。

【1709】図152はラッチーロジックラッチ表示を示しており、これが通常使用される配置である。しかしながら、多くのステージと一緒に置かれる時、「ステージ」をラッチーラッチーロジック（多くの技術者にとって良く知られているモード）と考えることも有効である。ラッチーロジックラッチ配置の使用は全てのブロック間通信が、ブロックを送る際にも受け取る際にも、ロジックを間に挟まずに、ラッチ対ラッチであるようにできる。

【1710】再び図152において、ロジックブロックを取り除き、データと有効信号を直接ラッチ間に接続し out valid と out accept がゲートされるのと同様に、入力上の NOR ゲートに直接ラッチさ

れた in valid を in accept ラッチに接続することにより、簡単な2線式インターフェース FIFO ステージが構築できる。データと有効信号は次に対応するアクセプト信号が高い時に伝搬する。図示した方法で in valid を out accept reg で OR ینگすることにより、データは out accept reg が低くても、in valid が低い場合にアクセプトされる。この方法で、ストール（低い信号をアクセプトする）が発生する時はいつでも、gaps（有効なビットが低いデータ）がパイプラインから取り除かれる。

【1711】図152に示すように、ロジックブロックを挿入し、in accept と out valid はデータもしくはブロックのステートに従属できる。図示した配置では、ブロック内のあらゆるステートが、ph1によって可能化されるマスターと、ph0によって可能化されるスレーブを備えた、マスター・スレーブ装置内に保持されることが標準である。

【1712】B.11.4 「スノーパブロック」
スノーパブロックはチップ内の様々なポイントにおいてマイクロプロセッサインターフェースを介してデータストリームへのアクセスを可能にする。スノーパブロックには2つのタイプがある。普通のスノーパはクロックが直接制御され得るテストモードにおいてのみアクセス可

能である。「スーパースノーバ」は、クロックが動いている間もアクセス可能であり、マイクロプロセッサバスからの非同期データを内部チップクロックに同期化させる回路を包含している。表230は時間デコード内の全

てのスノーバのロケーションとタイプを記載している。

【1713】

【表230】

ロケーション	タイプ
addrgen/vec pipe/snoopz31	スノーバ
addrgen/cnt pipe/midsnp	スノーバ
addrgen/cnt pipe/endsnp	スノーバ
addrgen/predread/snoopz44	スノーバ
addrgen/ip wrt2/superz10	スーパースノーバ
addrgen/ip rd2/superz10	スーパースノーバ
dramx/dramif/ifsnoops/snoopz15 (fsnp)	スノーバ
dramx/dramif/ifsnoops/snoopz15 (bsnp)	スノーバ
dramx/dramif/ifsnoops/superz9	スーパースノーバ
wruder/superz9	スーパースノーバ
pflts/fwdtflt/dimbuff/snoopk13	スノーバ
pflts/bwdtflt/dimbuff/snoopk13	スノーバ
pflts/snoopz9	スノーバ

表B.11.2 時間デコード内のスノーバ

両スノーバの使用に関する詳細はテストセクションに記載する。JTAGインターフェースのオペレーションの詳細はJTAG文書に記載する。

【1714】セクション B.12 「機能ブロック」 B.12.1 「トップフォーク」

本発明によれば、トップフォークは2つの異なる機能を果たす。第1に、データストリームを2つの異なるストリームに分岐する：1つはアドレス発生器に、他の1つをFIFOに、第2に、チップを配置できるように、チップのスターティング手段及びストップピング手段を提供する。

【1715】構成要素のフォーク部分の局面は非常に簡単である。同じデータがアドレス発生器とFIFOに表示され、アクセプトが前のステージに送り返される前に両ブロックによってアクセプトされていなければならない。このように、フォークの2つのブランチのvalidsが他のブランチからのアクセプトに従属する。チップが停止状態にある場合、両ブランチに対するvalidsは低く保たれる。チップは配置ビットが高く設定されるまで、in acceptが低く保たれる状態でパワーアップする。これはユーザーがチップを配置してしまうまで如何なるデータもアクセプトされないことを保証する。ユーザーが他の時にチップを配置する必要がある場合、ユーザーは配置ビットを設定し、チップが現在のストリームを完了してしまうまで待たなければならない。ストップピングプロセスは次の通りである：

1) 配置ビットが設定されていれば、トップフォークによってフラッシュトークンが検出された後、如何なるデ

ータもアクセプトされない。

【1716】2) フラッシュトークンがリードラダーに到達する時には、チップはストリームのプロセッシングを完了しているであろう。これは信号seq doneを高くさせる。

【1717】3) seq doneが高くなると、マイクロプロセッサが読むことができるイベントビットを設定する。イベント信号はイベントブロックによってマスクされ得る。

【1718】B.12.2 「アドレス発生器」

本発明では、アドレス発生器(addrgen)はフレーム内のブロック数をカウントし、DRAMデータ伝送用の正しいアドレスシーケンスを発生させる責任がある。アドレス発生器の入力は(トップフォークを介して)トークン入力ポートからのトークンストリームであり、DRAMインターフェースに対するその出力は、リクエスト/アクセラレーションプロトコルにより制御されるアドレス及び他の情報から成る。

【1719】アドレス発生器の基本的なセクションは以下の通りである：

- ・トークンデコード
- ・ブロックカウンティング及びDRAMブロックアドレスの発生
- ・モーションベクトルデータのアドレスオフセットへの変換
- ・予測伝送用のリクエスト及びアドレス発生器
- ・再配置リードアドレス発生器
- ・ライトアドレス発生器

B. 12. 2. 1 「トークンデコード (tokdec)」

トークンデコーダにおいて、コーディングスタンダードに連合するトークン、フレーム、ブロック情報、及びモーションベクトルがデコードされる。ストリームから抽出される情報は一連のレジスタに記憶され、それはup iを介してアクセスされ得る。データトークンヘッダの検出は次のブロックに合図され、ブロックカウンティング及びアドレス発生を可能化する。JPEGを動かしている時は何も起こらない。

【1720】デコードされるトークンリスト

- ・コーディングスタンダード
- ・データ
- ・DEFINE MAX SAMPLING
- ・DEFINE SAMPLING
- ・HORIZONTAL MBS
- ・MVD BACKWARDS
- ・MVD FORWARDS
- ・ピクチャスタート
- ・ピクチャタイプ
- ・予測モード

このブロックは更にリクエスト発生器からの情報を組み合わせて、フレームポインタのトグルリングを制御し、入力ストリームをストール (失速) させる。新しいフレームが (ピクチャスタートトークンの形態で) 入力に現れると、ストリームはストールされるが、前のフレームに関連するライトバックもしくは再配置リードは不完全である。

【1721】B. 12. 2. 2 「マクロブロックカウンタ (mbblkcnt)」

本発明のマクロブロックカウンタはフレーム内のマクロブロックの水平位置及び垂直位置を指す4つの基本的カウンタから成る。時間の始まりにおいて、また各ピクチャスタートに際して、全てのカウンタは0に設定される。データトークンヘッダが到着すると、カウンタは増分し、トークンヘッダ及びフレーム構造内の色成分ナンバーに従ってリセットされる。このフレーム構造はトークンデコーダ内のサンプリングレジスタによって説明される。

【1722】所定の色成分のために、カウンティングは次のように進行する。水平ブロックカウンタはマクロブロックの幅に達するまで、同じ成分の新しいデータトークンの各々に増分され、その後リセットする。垂直ブロックカウンタはマクロブロックの高さに達するまで、このリセットによって増分され、その後リセットする。これが発生すると、次の色成分を待つ。従って、このシーケンスはマクロブロック - おそらく各成分によって異なるであろう、マクロブロックの水平と垂直のサイズにおける成分の各々に対して繰り返される。ある成分に対して、期待されるより少ないブロックが受け取られた場

合でも、カウンタは誤差なしに次の成分へと進むであろう。

【1723】データトークンの色成分が期待されたバリュより少ない場合、水平マクロブロックカウンタが増分される。(所定の色成分に対して期待された以上のブロック数が現れた時にも、カウンタはより高い成分インデックスを期待しているので、こうしたことが発生するであろう。) この水平カウンタはカウンタがマクロブロック内のピクチャ幅に達する時にリセットされる。この

10 リセットは垂直マクロブロックカウンタを増分させる。

【1724】更に、H. 261 CIFフォーマット内のマクロブロックをカウントする能力がある。この場合、マクロブロックとブロック・グループと呼ばれるピクチャとの間の特別レベルの階層がある。これは11マクロブロック幅と3マクロブロックの深さであり、ピクチャは常に2グループ幅である。トークンデコーダはピクチャタイプトークンからCIFビットを引出し、これをマクロブロックカウンタに送り、ブロック・グループをカウントするよう指示する。成分毎にブロック数が少ないか、もしくは多すぎる場合、上記のような反応が引き起こされるであろう。

20

【1725】B. 12. 2. 3 「ブロック計算 (blkcalc)」

ブロック計算はマクロブロック及びマクロブロック内ブロックの座標をピクチャ内のブロックの位置用座標に変換する、つまり階層レベルをノックアウトする。もちろん、これは異なる色成分のサンプリング率を考慮しなければならない。

B. 12. 2. 4 「ベースブロックアドレス (bsblkadr)」

30 blkcalcからの情報は色成分オフセットと共に、線形DRAMアドレス空間内のブロックアドレスを計算するために使用される。本質的に、所定の色成分のために、線形ブロックアドレスは垂直ブロック数xピクチャ幅+水平ブロック数である。これはベースブロックアドレスを形成するために色成分オフセットに加算される。

【1726】B. 12. 2. 5 「ベクトルオフセット (vec pipe)」

トークンデコーダによって表示されるモーションベクトル情報は、水平及び垂直のピクセルオフセット座標の形態である。つまり、フォワード及びバックワードベクトルの各々のために、それが予測されているブロックに形成されるブロックから半ピクセルの転置を与える (x、y) がある。これらの座標は正または負であってよいことに注意。それらはまず各色成分のサンプリングに従って概算され、ブロック及び新しいピクセルオフセット座標を形成するために使用される。図154において、陰部分が形成されつつあるブロックを表す。点線の輪郭はそこからそれが来ることが予測されているブロックである。大きな矢印はブロックオフセット-予測ブロックの

50

オリジンを含むDRAMブロックへの水平及び垂直のベクトル、この場合(1、4)を示す。小さな矢印は新しいピクセルオフセット—そのDRAMブロック内の予測ブロックオリジンの位置を示す。DRAMブロックが8×8バイトであるので、ピクセルオフセットは(7、2)であるように見える。

【1727】乗算器アレイvmarrlaは次にブロックベクトルオフセットを線形ベクトルオフセットに変換する。ピクセル情報は(x、y)座標(pix info)として予測リクエスト発生器に送られる。

【1728】B. 12. 2. 6 「予測リクエスト」
フレームポインタ、ベースブロックアドレス、及びベクトルオフセットは加算されて、DRAM(Inblkad3)から引き出されるアドレスを形成する。ピクセルオフセットが0であれば、1つのリクエストだけが発せられる。x、またはy次元のいずれかにおいてオフセットがあれば、2つのリクエスト—オリジナルブロックアドレス、及びすぐ右かあるいは真下のもののいずれか—が発せられる。x及びyの両方にオフセットがあれば、4つのリクエストが発せられる。チップクロック様式とDRAMインターフェースクロック様式間の同期は第1の加算(Inblkad3)と、適当なリクエストを発するステートマシンとの間で起きる。このように、ステートマシン(psgstate)はDRAMインターフェースクロックによってクロックされ、その走査されたエレメントはDRAMインターフェーススキャンチェーンの一部を形成する。

【1729】B. 12. 2. 7 「再配置リードリクエスト及びライトリクエスト」

ここではピクセルオフセットが含まれないので、各アドレスは関連フレームポインタにベースブロックアドレスを加算することにより形成される。再配置リードは予測及びデータが他のフレーム記憶装置に書き戻されるので、同じフレーム記憶装置を使用する。各ブロックはリード及びライトデータの伝送が対応するアドレスにおける予測伝送を遅らせる傾向があるので、アドレスを記憶するため短FIFOを具備する。(これはリード/ライトデータが予測データよりチップデータフローに沿ったストリームと相互作用するからである。)更に、各ブロックはチップクロックとDRAMインターフェースクロック間の同期化を含む。

【1730】B. 12. 2. 8 「オフセット」

DRAMは2つのフレーム記憶装置として配置され、各々が3つの色成分を含む。フレーム記憶装置ポインタ及び各フレーム内の色成分オフセットがupiを介してプログラムされなければならない。

【1731】B. 12. 2. 9 「スノーバ」

本発明では、スノーバは次のように配置される：

b1kcalcとbsblkadrの間—このインターフェースは水平と垂直のブロック座標、適切な色成分

オフセット、及び(その成分用の)ブロック内のピクチャ幅から成る。

【1732】・bsblkadrの後—ベースブロックアドレス。

【1733】・vec pipeの後—線形ブロックオフセット、予測モード、色成分及びH. 261オペレーションに関する情報と共に、ブロック内のピクセルオフセット。

【1734】・Inblkad3の後—「予測リクエスト」の項で説明したように、物理的ブロックアドレス
スーパーノバは、外部DRAMのテスト中に使用するため、再配置リード及びライトリクエスト発生器の中に置かれる。詳細についてはDRAMインターフェースセクションを参照。

【1735】B. 12. 2. 10 「走査」

addrgenブロックはそれ自体のスキャンチェーンを持っており、そのクロッキングはブロック自体のクロック発生器(adclkgen)により制御される。ブロックの後端にあるリクエスト発生器はDRAMインターフェースクロック様式の範囲内にあることに注意。

【1736】B. 12. 3 「予測フィルタ」

本発明による予測フィルタの全体的な構造は図155に示す。フォワード及びバックワードフィルタは同じものであり、MPEGフォワード/バックワード予測ブロックをフィルタリングする。H. 261モードではフォワードフィルタだけが使用される(バックワードフィルタのh261 on入力はH. 261ストリームがバックワード予測を含まないので、永久的に低くあるべきである)。全体的な予測フィルタブロックは2線式インターフェースステージのパイプラインで構成される。

【1737】B. 12. 3. 1 「予測フィルタ」

各予測フィルタは他の予測フィルタとは完全に独立して作用し、有効データがその入力に現れるとすぐにデータを処理する。図156から明らかなように、予測フィルタは4つの別個のブロックから成り、その内の2つは同じものである。これらのブロックのオペレーションについてはMPEG及びH. 261のオペレーションのために独立して説明する方がよいであろう。H. 261は最も複雑であるので、最初に説明する。

【1738】B. 12. 3. 1. 1 「H. 261オペレーション」

使用される1次元フィルタ式は以下の通りである：

$$F_i = (x_{i+1} + 2x_i + x_{i-1}) / 4$$

$$(i \leq i \leq 6)$$

$$F_i = x_i \quad (\text{その他の } i)$$

これはx予測フィルタにより8×8ブロックの各ローに適用され、y予測フィルタにより各カラムに適用される。これが達成されるメカニズムは図157に図示するが、それは基本的にpfieldの概略図を表す。フィルタは8つの2線式インターフェースパイプラインステ

ージから成る。ローの最初と最後のピクセルのために、レジスタAとCがリセットされ、データはレジスタB、D、Fを無変更のまま通過する（BとDの内容は0に加えられる）。B×2muxの制御はレジスタbの出力が1だけ左にシフトされるようにセットされる。このシフティングはどのイベントにおいても常にシフトされる1つの場所に加えられるものである。こうして、全ての値に4が（後にこれ以上が）掛けられる。他の全てのピクセルのために、 x_{i+1} がレジスタCにロードされ、 x_i がレジスタBに、そして x_{i-1} がレジスタAにロードされる。図157から解るように、その後H. 261フィルタ式が実行される。垂直フィルタリングが3つの水平グループにおいて遂行されるので（下記のディメンションバッファに関するノートを参照）、ローにおける最初と最後のピクセルを別個に処理する必要がない。ロー内のピクセルの制御及びカウンティングは各1次元フィルタに関連する制御ロジックによって遂行される。その結果が4で割られていないことに注意すべきである。演算上の精度が失われないように、水平及び垂直のフィルタリングが行われた後、予測フィルタアダー（セクション 20 B. 12. 4. 2）の入力において、16で割る（4だ

け右にシフトする）作業が行われる。レジスタDA、D、DFが制御情報をパイプラインに送る。これはh261 onとlast byteを含む。

【1739】予測フィルタ内に見つけられる他のブロックの内、フォーマッティングの機能は単にデータが正しいオーダーでx-フィルタに表示されることを確実にすることである。上記から解るように、これには単に3ステージのシフトレジスタが必要であり、第1のステージはレジスタCの入力に接続され、第2のステージはレジスタBに、第3のステージはレジスタAに接続される。

【1740】xフィルタとyフィルタの間で、ディメンションバッファがデータをバッファリングし、3つの垂直ピクセルのグループがy-フィルタに表示されるようにする。これら3つのグループはまだ水平に処理されるが、予測フィルタ内では如何なる転置も発生しない。図158に関連して、ピクセルがディメンションバッファから出力されるシーケンスを表231、表232に示す。

【1741】

【表231】

クロック	入力 ピクセル	出力 ピクセル	クロック	入力 ピクセル	出力ピクセル
1	0	55(a)	17	16	7
2	1	56	18	17	F (b) (0, 8, 15)
3	2	57	19	18	F (1, 9, 17)
4	3	58	20	19	F (2, 10, 18)
5	4	59	21	20	F (3, 11, 19)
6	5	60	22	21	F (4, 12, 20)
7	6	61	23	22	F (5, 13, 21)
8	7	62	24	23	F (6, 14, 22)
9	8	63	25	24	F (7, 15, 23)

注a. 前のブロックがなかった場合（あるいはブロック間に長いギャップがあった場合）、前のブロックもしくは無効なデータからのピクセルの最も小さいロー。

b. F(x)はH. 261フィルタ式における関数を示す。

表B. 12. 1 H. 261ディメンションバッファシーケンス (1/2)

【1742】

【表232】

クロック	入力 ピクセル	出力 ピクセル	クロック	入力 ピクセル	出力ピクセル
10	9	0	26	25	F (8,16,24)
11	10	1	27	26	F (9,17,25)
12	11	2	28	27	F (10,18,26)
13	12	3	29	28	F (11,19,27)
14	13	4	30	29	F (12,20,28)
15	14	5	31	30	F (13,21,29)
16	15	6	32	31	F (14,22,30)

表B.12.1 H.261ディメンションバッファシーケンス (2/2)

B.12.3.1.2 「MPEGオペレーション」

MPEGオペレーションの間に、予測フィルタは簡単な半ピクセル補間を行う：

$$F_i = (x_i + x_{i+1}) / 2$$

($0 \leq i \leq 8$, 半ピクセル)

$$F_i = x_i \quad (0 \leq i \leq 7, \text{整数ピクセル})$$

ル)

h261 on入力が低くなければ、これはデフォルト

フィルタオペレーションである。1次元フィルタへの

信号dimが低ければ、整数ピクセル補間が実行されるであろう。従って、h261 onが低く、xdim及びydimが低ければ、全てのピクセルはフィルタリングを行わずに直接送られる。1次元フィルタへのdim信号が高い時に、ロー（またはカラム）が8ピクセル幅（もしくはそれ以上）になることが明らかな必要条件である。これは表233において要約されている。

【1743】

【表233】

h261 on	xdim	ydim	関数
0	0	0	$F_i = x_i$
0	0	1	MPEG8x9ブロック
0	1	0	MPEG9x8ブロック
0	1	1	MPEG9x9ブロック
1	0	0	H.261ローパスフィルタ
1	0	1	非合法的
1	1	0	非合法的
1	1	1	非合法的

表B.12.2 1次元フィルタオペレーション

図157、「1次元予測フィルタ」において、1次元フィルタのオペレーションは、H.261のローにおける最初と最後のピクセルのためであると同様、MPEG

インターピクセルのためである。MPEG半ピクセルオペレーションのために、レジスタAは永久的にリセットされ、レジスタCの出力は1だけ左にシフトされる（レ

ジスタBの出力は常に1だけ左にシフトされる)。こうして、2個のクロックの後、レジスタFは(2B+2C)を含み、それは必要な結果の4倍であるが、これはxフィルタ及びyフィルタを通過した数が4だけ右にシフトされる予測フィルタアダーの入力において処理される。

【1744】フォーマッティング及びディメンションバッファの機能はMPEGにおいてもシンプルである。フォーマッティングは2個の有効なピクセルを集め、それらを半ピクセル補間のためにx-フィルタに送らなければならない；ディメンションバッファは1つのローをバッファすることだけが必要である。注目に値することは、データがx-フィルタを通過した後、フィルタリングオペレーションが9-ピクセルローを8-ピクセルローに変換するので、ローの中に8個のピクセルだけが存在することである。「失われた」ピクセルはデータストリーム内のギャップで置き換えられる。半ピクセル補間を遂行する時、x-フィルタは各ローの終わりに(各8個のピクセルの後に)ギャップを挿入し；y-フィルタはブロックの終わりに8個のギャップを挿入する。このことは、ブロックの終わりで、8個もしくは9個のギャップ・グループがデータトークンヘッダ、及びFIFOから来るストリーム内のデータトークン間の他のトークンと整列することから、重要である。これは9x9のブロックがフィルタリングされる時に発生する、チップを通じて最悪の場合を最小限に抑える。

【1745】B. 1.2. 3. 2 「予測フィルタアダー」

MPEGオペレーションの間に、初期のピクチャ、後期のピクチャ、または両者の平均を用いて予測が形成される。初期のフレームから形成される予測はフォワード予測と呼ばれ、後期のフレームから形成される予測はバックワード予測と呼ばれる。予測フィルタアダー(pfad)の機能は、どちらのフィルター済み予測値を使用するか(フォワード、バックワードもしくはその両方)、そしてフォワードまたはバックワードフィルター済み予測もしくは両者の平均のいずれを通過するかを決定することである(正の無限大に向かってラウンドされる)。

【1746】予測モードはブロック間で、つまりパワーアップ時に、もしくは現在の予測ブロックの最後のバイトを指示するfwd 1st byte及び/もしくはbwd 1st byte信号が活性になった後、変更できるだけである。現在のブロックがフォワード予測であれば、fwd 1st byteだけが調べられる。もしそれがバックワード予測であれば、bwd 1st byteだけが調べられる。もしそれが二方向性の予測であれば、fwd 1st byte及びbwd 1st byteが調べられる。

【1747】信号fwd oh及びbwd ohはどの

予測値を使用するかを決定する。随時、これらの信号の両方が活性であっても、もしくは両方共活性でなくともよい。スタートアップ時に、あるいは、ブロックの入力に有効なデータが存在しない時にギャップがあれば、ブロックはどちらの信号も活性でない時にステートに入る。

【1748】次のブロックのための予測モードを決定するのに2つの基準が使用される：フォワードブロックもしくはバックワードブロックのいずれが二方向性の予測ペアの一部であるかを指示する信号fwd ima twinとbwd ima twin、及びバスfwd p num[1:0]及びbwd p num[1:0]である。これらのバスは各々の新しい予測ブロックもしくは予測ブロックペアのために、1だけ増分する数字を含む。これらのブロックが必要であるのは、例えば、2つのフォワード予測ブロックとそれに続く二方向性の予測ブロックがある場合、第2のフォワード予測ブロックの前に予測フィルタアダーの入力に到達するように、DRAMインターフェースは充分前に二方向性予測のバックワードブロックを引き出すことができるからである。同様に、他のシーケンスのバックワード及びフォワード予測も予測フィルタアダーの入力においてシーケンスから出ることができる。このように、次の予測モードは次のように決定される：

1) 有効フォワードデータが存在し、fwd ima twinが高ければ、ブロックは有効バックワードデータがbwd ima twinセットと共に到着するまでストールし、それから各予測バリューペアを平均化するブロックを通過する。

【1749】2) 有効バックワードデータが存在し、bwd ima twinが高ければ、ブロックは有効フォワードデータがfwd ima twinセットと共に到着するまでストールし、それから上述のように進む。フォワード及びバックワードデータが共に有効であれば、ストールは行われない。

【1750】3) 有効フォワードデータは存在するが、fwd ima twinが設定されなければ、fwd p numが調べられる。これが(pred numに記憶されている)前の予測+1からの数字と等しければ、予測モードがフォワードに設定される。

【1751】3) 有効バックワードデータは存在するが、bwd ima twinが設定されなければ、bwd p numが調べられる。これが(pred numに記憶されている)前の予測+1からの数字と等しければ、予測モードがバックワードに設定される。

【1752】パイプライン内の1ステージ後方からのearly valid信号が使用され、新しいブロックからの最初のデータが到着する前に予測フィルタアダーモードを設定することができることに注意。これはパイプラインに如何なるストールも導入されないことを保証

する。

【1753】ima twin及びpred num信号はフィルタリングされたデータと共に、フォワード及びバックワード予測フィルタパイプラインに沿って送られない。これは以下の理由からである：

1) これらの信号は、fwd lst byte及び／もしくはbwd lst byteが有効である時にのみ調べられる。それにより、各予測フィルタにおいて約25の3ビットパイプラインステージを節約できる。

【1754】2) ブロック中を通じて信号は有効なままであるので、fwd lst byte及び／もしくはbwd lst byteが予測フィルタアダーに到着する時に、有効である。

【1755】3) 信号は、いずれにしてもデータが到着する1クロック前に調べられる。

【1756】B. 12. 4 「予測アダー及びFIFO」

予測アダー（パダー）は予測フィルタからのデータを誤差データに加算することにより、予測済みフレームを形成する。アドレス発生器、DRAMインターフェース及び予測フィルタを通る入力からのディレイを補償するため、誤差データはパダーに達する前に256ワードFIFO(s f i f o)を通過する。

【1757】コーディングスタンダードトークン、予測モードトークン、及びデータトークンは、いつ予測ブロックが形成されるかを決定するためにデコードされる。8ビットの予測データはデータトークン内の9ビットの2の補数誤差データに加算される。その結果は0～255の範囲に制限され、次のブロックに進む。このデータ制限はJPEGを含む全てのイントラコード化データにも適用されることに注意。

【1758】本発明の予測アダーは更に、FIFO及び予測フィルタから到着するデータにおける不整合を検出するためのメカニズムを含む。理論上は、フィルタから

のデータ量は、予測を含むFIFOからのデータトークン数に正確に対応していなければならない。重大な機能不全の場合、パダーがリカバリを試みる。

【1759】FIFO及びフィルタからのデータブロックの終わりがin extn及びf l l a s t入力によって、各々マークされる。フィルタデータの終わりがデータトークンの終了前に検出された場合、残りのトークンは変更されずに出力を続ける。他方、フィルタブロック…データトークンより長い場合、全ての過剰フィルタデータがアクセプトされ、捨てられるまで入力はストールされる。

【1760】トークン入力ポートからのデータを直接これらのブロックに送り、またトークン出力ポートにこれらの出力を直接送るようにチップが構成されるので、FIFOもしくは予測アダーのいずれにもスノープはない。

【1761】B. 12. 5 「ライト及びリードラダー」

B. 12. 5. 1 「ライトラダー(wrudde r)」

ライトラダーは予測アダーから来る全てのトークンをリードラダーに送る。更に、それはMPEGのIまたはPピクチャ内の全てのデータブロック、及びH. 261とDRAMインターフェース内の全てのデータブロックを送るので、アドレス発生器の制御下に外部フレーム記憶装置にそれらをライトバックすることができる。ライトバックデータはDRAMインターフェースに行く途中のスノープを通過するが、全ての基本的な機能性は1つの2線式インターフェースステージ内に包含される。

【1762】ライトラダーは以下のトークンをデコードする：

【1763】

【表234】

トークン名	ライトラダーの機能
CODING STANDARD	JPEGストリームのためにライトバックは禁じられる
PICTURE TYPE	ライトバックはBフレームではなく、IフレームとPフレームにおいてのみ発生する
DATA	DATAトークン内のデータだけがライトバックされる。

表B. 12. 3 ライトラダーによりデコードされるトークン

データトークンヘッダが検出された後、全てのデータバイトがDRAMインターフェースに出力される。データトークンの終わりは低くなるin extnによって検出され、これはフラッシュ信号をDRAMインターフェ

ーススイングバッファに送るようにする。通常のオペレーションでは、これはスイングバッファがとにかくスイングするであろうポイントと整列するであろうが、データトークンが64バイトのデータを含んでいなければ

これはリカバリメカニズムを提供する（但し、次の2～3の出力ピクチャが不正確であるかもしれない）。

【1764】B. 12. 5. 2 「リードラダー (r r u d d e r)」

本発明のリードラダーは次の3つの機能を持ち、主要な2つの機能はMPEGにおけるピクチャシーケンス再配置に関するものである：

1) 外部フレーム記憶装置からリードバックされたデータを正しい位置でトークンストリームに挿入すること。

【1765】2) Iピクチャ及びPピクチャ内のピクチャヘッダ情報を再配置すること。

【1766】3) フラッシュトークンを検出することにより、トークンストリームの終わりを検出すること（セクション B. 12. 1「トップフォーク」を参照）。

【1767】リードラダーの構造は図159に図示されている。全体のブロックは標準2線式インターフェース技術から作られる。入力インターフェースラッチ内のトークンはデコードされ、これらのデコードはブロックオペレーションを決定する：

【1768】

【表235】

トークン名	リードラダーの機能
FLUSH	トップフォークへの信号
CODING STANDARD	コーディングスタンダードがMPEGでなければ、再配置を禁じる。
SEQUENCE START	再配置されたシーケンスの最初のピクチャ用のリードバックデータは無効である。
PICTURE START	現在の出力FIFOをスワップしなければならないことを告げる信号（IまたはPピクチャ）。ピクチャヘッダトークンの最初のもの。
PICTURE END	ピクチャレイヤの上の全てのトークンは通過することが許される。
TEMPORAL REFERENCE	ピクチャヘッダトークンの二番目のもの。
PICTURE TYPE	ピクチャヘッダトークンの三番目のもの。
DATA	再配置の際に、DATAトークンの内容が再配置されたデータで置き換えられる。

表B. 12. 4 リードラダーによりデコードされるトークン

再配置機能はマイクロプロセッサインターフェースを介して始められるが、レジスタのステートに関わらず、コーディングスタンダードがMPEGでなければ禁じられる。同じMPIレジスタがアドレス発生器が再配置アドレスを発生しているかどうかを制御し、このように再配置はこのブロックからの出力である。リードラダーがどのように作用するかを理解するため、トークンのシーケンスが以下の通りであることを念頭において、入力と出力の制御ロジックを別々に考慮する：

- ・コーディングスタンダード
- ・シーケンススタート
- ・ピクチャスタート
- ・時間標準
- ・ピクチャタイプ
- ・データトークンと他のトークンを含むピクチャ
- ・ピクチャエンド
- ・...

・ピクチャスタート

B. 12. 5. 2. 1 「入力制御ロジック」

パワーアップから、全てのトークンは、IもしくはPピクチャ用の最初のピクチャタイプトークンに遭遇するまで、FIFO1（現在の入力FIFOと呼ばれる）に進む。次に、FIFO2が現在の入力FIFOになり、IもしくはPピクチャ用の次のピクチャタイプに遭遇するまで全ての入力がそれに向けられ、FIFO1が再び現在の入力FIFOになる。I及びPピクチャ内で、データトークンを除き、ピクチャタイプとピクチャエンドの間の全てのトークンが捨てられる。これは何の意味も持たないであろう再配置されたストリームにおいて、モーションベクトルが間違ったピクチャと連合することを防止する。

【1769】3ビットコードがトークンストリームと共にFIFOに挿入され、特定のトークンヘッダの存在を指示する。これによりFIFOの出力に関するトークンデコーディングを行う必要がなくなる。

【1770】B. 12. 5. 2. 2 「出力制御ロジック」

パワーアップから、ピクチャスタートコードに遭遇するまで、トークンはFIFO1（現在の出力FIFOと呼

ばれる) アクセプトされ、その後FIFO2が現在のFIFOになる。セクション B. 12. 5. 2. 1において、この段階で3つのピクチャヘッダトークン、ピクチャスタート、時間標準、ピクチャスタートがFIFO1に保持されていることが解る。現在の出力FIFOはIもしくはPフレームにおいてピクチャスタートコードと遭遇する度にスワップされる。従って、3つのピクチャヘッダトークンは次のIもしくはPフレームまで記憶され、次のIもしくはPフレームでは、それらは正確に再配置されたデータと連合するようになるであろう。B 10
ピクチャは再配置されず、如何なるトークンも捨てられずに通り過ぎる。ピクチャエンドを含む最初のピクチャ内の全てのトークンが捨てられる。

【1771】I及びPピクチャの間に、トークンストリームのデータトークンに含まれるデータはDRAMインターフェースからの再配置されたデータで置き換えられる。最初のピクチャの間、「再配置された」データは再配置されたデータ入力に存在する。なぜなら、アドレス発生器がDRAMインターフェースにそれを引き出すように要請するからである。これはハッシュと考えられ、20
捨てられる。

【1772】セクション B. 13 「DRAMインターフェース」

B. 13. 1 「展望」

本発明では、空間デコーダ、時間デコーダ及びビデオフォーマットは各々その特別なチップのためにDRAMインターフェースブロックを具備する。3つ全ての装置において、DRAMインターフェースの機能は、チップからのデータを外部DRAMに伝送し、アドレス発生器により供給されるブロックアドレスを介して、外部 30
DRAMからのデータをチップ内へと伝送することである。

【1773】典型的に、DRAMインターフェースはアドレス発生器、及びデータがそれを通して送られる様々なブロックのクロックに非同期的であるクロックから操作する。しかしながら、クロックはほぼ同じ周波数で操作するので、この非同期化は容易に処理される。

【1774】通常、データはDRAMインターフェースと64バイトのブロック内の残りのチップとの間に伝送される(唯一の例外は、時間デコーダ内の予測データである)。伝送は「スイングバッファ」として知られる装置によって行われる。これは本質的にダブルバッファされた配置において操作される一組のRAMであり、DRAMインターフェースが1つのRAMを結めるか空にしている間に、別のチップ部分が他のRAMを空にするか、結めている。アドレス発生器からアドレスを運ぶ別のバスが各々のスイングバッファと連合する。

【1775】各チップは4つのスイングバッファを持つが、これらのスイングバッファの機能は各々の場合で異なる。空間デコーダにおいて、コード化データをDR 50

Mに伝送するために1つのスイングバッファが使用され、DRAMからコード化データを読むために別のスイングバッファが使用され、トークン化データをDRAMに伝送するために3番目のスイングバッファが、またDRAMからトークン化データを読むために4番目のスイングバッファが使用される。時間デコーダにおいては、イントラもしくは予測されたピクチャデータをDRAMに書き込むために1つのスイングバッファが使用され、DRAMからイントラもしくは予測されたピクチャデータを読むために2番目のスイングバッファが使用され、他の2つはフォワード及びバックワード予測データを読むために使用される。ビデオフォーマットにおいては、1つのスイングバッファがDRAMにデータを伝送するために使用され、他の3つがDRAMからデータを読むために使用されるが、それは各々輝度(Y)、及び赤と青の色差データ(各々Cr及びCb)である。DRAMインターフェースの一般的な特徴のオペレーションについては、空間デコーダ文書において記載している。次のセクションは時間デコーダに特有の特徴について説明する。

【1776】B. 13. 2 「時間デコーダDRAMインターフェース」

セクション B. 13. 1で述べたように、時間デコーダは4つのスイングバッファを持ち：2つはデコード化イントラ及び予測化(I及びP)ピクチャデータをリード/ライトするために使用され、これらは上述したように操作する。他の2つは予測データを引き出すために使用される。

【1777】一般に、予測データはx及びyにおけるモーションベクトルによって指定されるように処理されるブロックの位置からオフセットしているであろう。このように、引き出されるべきデータブロックは一般にそれがエンコードされた(そしてDRAMに書き込まれた)ようにデータのブロック境界に対応してはいないであろう。これは図160に図示しているが、陰領域は形成されつつあるブロックを表している。点線の輪郭はそこからそれが予測されているブロックを示す。アドレス発生器は大きな矢印で示すように、ブロックオフセット(ブロックの全数)に対するモーションベクトルによって指定されるアドレスを変換し、小さな矢印で示すように、ピクセルオフセットに対するモーションベクトルによって指定されるアドレスを変換する。

【1778】アドレス発生器において、フレームポインタ、ベースブロックアドレス、及びベクトルオフセットはDRAMから引き出されるブロックのアドレスを形成するために加算される。ピクセルオフセットが0であれば、1つのリクエストだけが発せられる。xまたはy次元のいずれかにおいてオフセットがあれば、2つのリクエスト-オリジナルブロックアドレス、及びすぐ右かあるいは真下のもののいずれかが発せられる。x及びy

の両方にオフセットがあれば、4つのリクエストが発せられる。引き出されるべき各々のブロックのために、アドレス発生器はスタート/ストップアドレスのパラメータを計算し、それらをDRAMインターフェースに送る。これらのスタート/ストップアドレスの使用については、下記において概略するように実例によって説明するのが良いであろう。

【1779】図161において陰領域で示すように、

(1, 1)のピクセルオフセットを考えてみよう。アドレス発生器は、図においてAからDのラベルが貼られた4つのリクエストをする。解決すべき問題は求められるローアドレスのシーケンスを如何にすばやく提供するかである。解決法は「スタート/ストップ」技術を使用することであり、これについて説明する。

【1780】図161のブロックAを考えてみよう。リーディングは位置(1, 1)でスタートし、位置(7, 7)で終了しなければならない。しばらく、1バイトが一度に(つまり、8ビットのDRAMインターフェース)読まれていると仮定する。座標ペアのxバリューはアドレスの3LSBを形成し、yバリューは3つのMSBを形成する。x及びyのスタートバリューは共に1であり、アドレス9を与える。データはこのアドレスから読まれ、xバリューは増分される。プロセスはxバリューがそのストップバリューに達するまで繰り返される。この時点で、yバリューは1だけ増分され、xスタートバリューは再ロードされ、アドレス17を与える。データの各バイトが読まれるにつれて、xバリューはストップバリューに達するまで再び増分される。xとyバリューが共にそのストップバリューに達するまで、プロセスが繰り返される。こうして、9、10、11、12、13、14、15、17、...

3、14、15、17、... 23、25、...、31、33、...、57、...、63のアドレスシーケンスが生成される。

【1781】同様に、ブロックB用のスタート/ストップ座標は：(1, 0)及び(7, 0)であり、ブロックC用には：(0, 1)及び(0, 7)であり、ブロックD用には：(0, 0)及び(0, 0)である。

【1782】次の問題はこのデータをどこに書くべきかということである。明らかに、ブロックAを見ると、アドレス9から読まれるデータはスイングバッファのアドレス0に書かれるべきであり、アドレス10から読まれるデータはスイングバッファのアドレス15に書かれるべきである等である。同様に、ブロックBのアドレス8から読まれるデータはスイングバッファのアドレス15に書かれるべきであり、アドレス16から読まれるデータはスイングバッファのアドレス15に書かれるべきである。この機能は下記に概略するように、非常に簡単な実行であることが解る。

【1783】ブロックAを考えてみよう。リーディングのスタートにおいて、スイングバッファアドレスレジスタには、ストップバリューの逆がロードされ、y逆ストップバリューは3つのMSBを形成し、x逆ストップバリューは3つのLSBを形成する。この場合、DRAMインターフェースが外部DRAM内のアドレス9を読んでいる間、スイングバッファアドレスは0である。その後、外部DRAMアドレスレジスタが増分されるにつれて、表236に図示するように、スイングバッファアドレスレジスタが増分される：

【1784】

【表236】

外部DRAMアドレス	スイングバッファアドレス
9=y-start, x-start	0=y-stop, x-stop
10	1
11	2
15	6
17=y+1, x-start	8=y+1, x-stop
18	9

外部DRAMアドレス (バイナリー)	スイングバッファアドレス (バイナリー)
001 001	000 000
111 110	000 001
001 011	000 010
001 111	000 110
010 001	001 000
010 010	001 001

表B. 13. 1 予測アドレッシングの実例

今までの議論は8ビットのDRAMインターフェースを中心にしていた。16または32ビットのインターフェースの場合、少しばかり修正をしなければならない。ま

ず、ピクセルオフセットベクトルはそれが16または32ビットバイナリーを指すように「クリップ」されなければならない。今まで使用してきた例において、ブロッ

クA用に、最初のDRAMリードはアドレス0を指し、アドレス0～3の中のデータが読まれるであろう。次に、望まれていないデータは捨てられなければならない。これは全てのデータをスイングバッファに書き込み（これは今では8ビットの場合に必要なものより物理的に大きくなっているにちがいない）、オフセットで読むことによって遂行される。MPEG半ピクセル補間を行う時、x及びyの中の9バイトがDRAMインターフェースから読まれなければならない。この場合、アドレス発生器は適切なスタートとストップアドレスを提供し、DRAMインターフェース内の幾つかの追加ロジックが使用されるが、DRAMインターフェースが操作する方法には基本的な変更はない。

【1785】時間デコーダDRAMインターフェースに関して注意すべき最後のポイントは、データに関してどのようなプロセッシングが求められているかを指示するため、予測フィルタに追加情報を提供しなければならないことである。これには次の点が含まれる：

- ・(64、72または81バイトの) 伝送の最後のバイトを指示する「最後のバイト」信号
- ・H. 261フラグ
- ・二方向性の予測フラグ
- ・ブロックの次元を指示する2ビット (x及びyにおける8または9バイト)
- ・ブロックのオーダーを指示するための2ビットのナンバー

最後のバイトフラグはデータがスイングバッファから読まれる時に生成される。他の信号はアドレス発生器から引き出され、予測フィルタブロックによってスイングバッファから読まれるにつれて、それらが正しいデータブロックと連合するように、DRAMインターフェースを通してパイプ接続される。

【1786】セクション B. 14 「UPI文書化」 B. 14. 1 「序文」

本文書は本発明によるマイクロプロセッサインターフェースのオペレーションの正しい認識を読者に与えることを目的とする。インターフェースは基本的に空間デコーダ及び時間デコーダに関するものと同じであり、唯一の違いはアドレスラインの数である。

【1787】ここに記載されるロジックは純粋にマイクロプロセッサ内部ロジックである。関連概略図は以下の通りである：

UPI
UPI101
UPI102
DINLOGIC
DINCELL
UPIN
TDET
MONOVRLP

WRTGEN
READGEN
VREFCKT

回路UPI、UPI101、UPI102は、UPI01が7ビットのアドレス入力を持ち、8番目のビットが大地にハードワイヤードされる一方、他の2つが8ビットのアドレス入力を持つこと以外は総て同じである。

入力信号

ここで説明する信号は (UPIに関して定義された) UPIモジュールに対する全ての入力及び出力のリストであり、これらの信号のソースもしくは目的地を詳細に記す注を付記している：NOTRSTInputGlobalチップリセット、活性低、パッド入力ドライバから。

【1788】E1InputEnable信号1、活性低、パッド入力ドライバから (Schmitt)。

【1789】E2InputEnable信号2、活性低、パッド入力ドライバから (Schmitt)。

【1790】RNOTWInputRead not Write信号、パッド入力ドライバから (Schmitt)。

【1791】ADDRIN[7:0] Input Address bus信号、パッド入力ドライバから (Schmitt)。

【1792】NOTDIN[7:0] Input Input data bus、二方向性マイクロプロセッサデータピンの入力パッドドライバから (TTLin)。

【1793】INT RNOTWOutputThe Read not Write信号、マイクロプロセッサインターフェースによりアクセスされる内部回路へ (メモリーマップ参照)。

【1794】INT ADDR[7:0] Output The Internal Address Bus、マイクロプロセッサインターフェースによりアクセスされる全ての回路へ (メモリーマップ参照)。

【1795】INTDBUS[7:0] Input/Output The Internal Data bus、マイクロプロセッサインターフェースによりアクセスされる全ての回路へ (メモリーマップ参照) 及びマイクロプロセッサデータ出力パッドへ。内部データバスはチップのピン上のデータと逆であるデータを伝送する。

【1796】READ STROutputAnはデバイスメモリーマップ内のロケーションの読み取りを指示する内部タイミング信号である。

【1797】WRITE STROutputAnは内部メモリーマップ内のロケーションの書き込みを指示する内部信号である。

【1798】TRISTATEDPADOutputAnはそれらがトライステートであるべきことを指示するマイクロプロセッサデータ出力パッドに接続する内部信

号である。一般的なコメント：UPI概略図は6個の小さなモジュールで構成される：NONOVRLP、UPIN、DINLOGIC、VREFCKT、READGEN、WRTGEN。信号の全体的なリストから注意すべきことは、チップ上の他の全てのタイミング信号と同期するマイクロプロセッサバスタイミング信号以外のマイクロプロセッサインターフェースと連合するクロック信号はないということである。従って、マイクロプロセッサのオペレーション、及び外部制御により強制することが出来るもの以外の残りのデバイスのオペレーションとの間に、如何なるタイミング関係も仮定されるべきではない。例えば、テストシステム上でのマイクロプロセッサインターフェースへのアクセシングに対して、外部システムクロックのストップング。

【1799】UPIにおいてクロックを持たない他の密接な関係は、幾つかの内部タイミングがセルフタイムであることである。つまり、ある信号のディレイはUPIブロックに対して内部的に制御される。

【1800】UPIの全体的な機能は、外部世界からアドレス、データ、イネーブル、及びリード/ライト信号を取り、それらが内部回路を正しく動かすことができるようにそれらをフォーマットすることである。メモリーマップに対するアクセスを定義する内部信号は、INT RNOTW INT ADDR [. . .]、INTD BUS [. . .]、及びREAD STRとWRITE STRである。これらの信号のタイミング関係は、リードサイクルとライトサイクル用に下記に示す。データシート定義及び下記の線図は常にチップイネーブルサイクルを示しているが、回路オペレーションはイネーブルが低く保持され、アドレスは連続的リードまたはライトオペレーションを行うために循環され得るようになってい

【1801】更に、INT RNOTW及びREAD STR、WRITE STRの存在は幾らかの冗長度を反映している。それは内部回路が別々のREAD STRとWRITE STRのいずれかを使用する（そしてINT RNOTWを無視する）、あるいはINT RNOTWと別個のストロブ信号（ストロブ信号はREAD STR及びWRITE STRのORから引き出される）を使用することができるようにする。

【1802】内部データバスはリードサイクルの間高くプレチャージされ、それは更に内部データバスが駆動されていない時、延長された期間の間、0XFF状態にデフォルトするように抵抗型プルアップを持つ。内部データバスはピン上のデータの逆であるので、これはそれらが可能化される時、外部ピン上の0x00に翻訳する。これは、メモリーマップ内のホールであるレジスタもしくはレジスタのビットに外部サイクルがアクセスする場合、出力データは限定され、slowであることを意

味する。

【1803】回路の詳細：

UPIN -

この回路は全体的な変化検出ブロックである。それは1ビットの変化検出回路であるTDETと呼ばれるサブ回路を含む。UPINは各アドレスビットのためにTDETモジュールを、そして各々のイネーブル信号のためにrnotwを持つ。UPINは更に、変化検出回路の出力を共にゲートするためにある種の結合ロジックを含む。このゲーティングは以下の信号を発生させる：

TRAN - 入力信号の1つへの遷移を指示する、そして

UPD DONE - 遷移が完了し、サイクルが実行され得ることを指示する。

【1804】CHIP EN - チップが選択されたことを指示する。

【1805】TDET -

これは1ビット変化検出回路である。2つのラッチと2つの専用ORゲートから成る。第1のラッチは信号SAMPLEによってクロックされ、第2のラッチは信号UPDATEによってクロックされる。これら2つのノンオーバーラッピング信号はモジュールNONOVRLPから来る。一般的なオペレーションは入力遷移がCHANGEを生じさせ、次にそれがSAMPLEを生じさせるように行われる。SAMPLEが高い間に全ての入力変化がアクセプトされ、入力変化が止まる時に、CHANGEが低くなり、SAMPLEが低くなり、それはUPDATEを高くさせ、それは次にデータを出力ラッチに伝送し、UPD DONEを指示する。

【1806】NONOVRLP -

この回路は基本的に、TRANを入力し、SAMPLEとUPDATEを発生させるノンオーバーラッピングクロック発生器である。UPDATEの出力上の外部ゲーティングはライトパルスが完了してしまうまで、UPDATEが高くなるのを停止させる。

【1807】DINLOGIC -

このモジュールはデータ入力回路DINCELLの8段階と、TRISTATEPAD信号を駆動させる幾つかのゲーティングで構成される。これは出力データポートがEnable1が低く、Enable2が低く、Rnotwが高く、内部read strが高い場合にのみ駆動するであろうことを指示する。

【1808】DINCELL -

この回路はデータ入力ラッチと内部データバスを駆動させるトライステートドライバから成る。信号DATAHOLDが高く、Enable1とEnable2の両方が低い時に、入力パッドからのデータがラッチされる。トライステートドライバは内部信号INT RNOTWが低い時にはいつでも、内部データバスを駆動する。内部データバスはトランジスタをプレチャージし、バスブ

ルアップもこのモジュールに含まれる。

【1809】WRTGEN-

このモジュールはWRITE STRと、データラッチのためにラッチ信号DATAHOLDを発生させる。ライトストローブはセルフタイム信号であるが、セルフタイムディレイはVREFCKTにおいて定義される。タイミング回路RESETWRITEからの出力はWRITE STR信号を終了させるために使用される。レジスタに書き込む実際のライトパルスはアクセスサイクルが終わった後にのみ生じることには注意すべきである。これはチップに対するデータ入力サイクルのバックエッジの上でのみサンプルされるからである。従って、データは通常のアクセスサイクルが終了した後でのみ有効である。

【1810】READGEN-

この回路は、その名前が示す通り、READ STRを発生させ、更に、内部データバスをプレチャージするために使用されるPRECH信号を発生させる。PRECH信号もセルフタイム信号であり、その期間はVREFCKT、及び内部データバスの電圧に依存する。READ STRはセルフタイムではないが、プレチャージ期間の終わりからサイクルの終わりまで続く。プレチャージ回路はインバータを使用するが、その伝送特徴はそれらが逆相にする前に供給電圧のほぼ75%の電圧を必要とするように、バイアスされる。この回路はREAD STRが始まる前に、内部バスが正しくプレチャージされることを保証する。内部バスが既にプレチャージされている場合、ゼロ幅になる傾向があるPRECHパルスを停止させるために、タイミング回路は信号RESET READを介して、最低の幅を保証する。

【1811】VREFCKT-

VREFCKTはインターフェースのセルフタイミングを制御する唯一の回路である。両ディレイ、WRITE STRの1/Wid th及びPRECHの2/Wid thはPトランジスタを通る電流によって制御される。このPトランジスタのゲートは信号VREFによって制御され、この電圧は25kΩの拡散型トランジスタによって設定される。

【1812】セクションC. 1 「展望」

C. 1. 1 「序文」

本発明によるイメージフォーマッティング部の構造を図164に示す。1つはライティング用、1つはリーディング用に2個のアドレス発生器、2個のアドレス発生器を管理し、フレーム率変換を提供するバッファマネージャ、垂直及び水平のアンサンブラを含むデータプロセッシングパイプライン、色空間変換及びガンマ補正、そしてプロセッシングパイプラインの出力を調整する最終制御ブロックがある。

【1813】C. 1. 2 「バッファマネージャ」

イメージフォーマッティング部の入力に到達するトークン

ンはFIFOにおいて緩衝され、バッファマネージャに伝送される。このブロックは新しいピクチャの到着を検出し、各ピクチャを記憶するべきバッファの利用可能性を判断する。利用できるバッファがあれば、それが到着するピクチャに割り当てられ、そのインデックスがライトアドレス発生器に伝送される。利用できるバッファがなければ、入ってくるピクチャはいずれかのバッファが利用できるようになるまでストールされる。全てのトークンはライトアドレス発生器に送られる。

【1814】リードアドレス発生器がVSYNC信号をディスプレイシステムから受け取る度に、新しいディスプレイバッファインデックスのためにバッファマネージャに対してリクエストが為される。完全なピクチャデータを含むバッファがあれば、そしてそのピクチャが表示のための準備が整ったと思われる場合、バッファのインデックスがディスプレイアドレス発生器に送られる。そうでなければ、バッファマネージャは表示されるべき最後のバッファインデックスを送る。スタートアップ時に、最初のバッファが一杯になるまでインデックスとして0が送られる。(各ピクチャが入力される時に計算される)ピクチャの数が、エンコーディングフレーム率が与えられるディスプレイ(提示数)で予測されるピクチャ数より大きい場合、ピクチャは表示のための準備が整っている。予測される数はピクチャクロックパルスをカウントすることにより決定され、その場合ピクチャクロックはクロック分配器により局部的に、あるいは外部で発生させることができる。この技術はフレーム率変換(例えば、2-3プルダウン)を可能にする。

【1815】外部DRAMはバッファのために使用され、その数は2個であっても3個であってもよい。フレーム率変換が行われることになっている場合は3個必要である。

【1816】C. 1. 3 「ライトアドレス発生器」

ライトアドレス発生器はバッファマネージャからトークンを受け取り、各々の新しいデータトークンの到着を検出する。各データトークンが到着するにつれて、アドレス発生器は到着するブロックを記憶するためのDRAMインターフェース用に新しいアドレスを計算する。生のデータはその後DRAMインターフェースに送られ、そこでシングバッファに書き込まれる。DRAMアドレスはブロックアドレス、及びDRAM内のピクチャまたはブロックのラスタとして組織されるピクチャである。しかしながら、入ってくるピクチャデータは実際に組織化されるマイクロプロセッサのシーケンスであるので、アドレス発生アルゴリズムはマイクロプロセッサ内のブロックの下位ローのための(ブロック内)ライン幅オフセットを考慮しなければならない。

【1817】バッファマネージャにより提供される到着バッファインデックスは記憶されるピクチャ全体のためにアドレスオフセットとして使用される。更に、各成分

は指定されたバッファ内の別のエリアに記憶されるので、成分オフセットは計算の中でも使用される。

【1818】C. 1. 4 「リードアドレス発生器」

リードアドレス発生器 (dispaddr) はトークンの受取及び発生を行わず、アドレスだけを発生させる。VSYNCに答え、アドレス発生器はfieldinfo、readstart、syncmode、及びlsbinvertに応じて、バッファマネージャからのバッファインデックスを要請する。インデックスを受け取ると、3セットのアドレス、各成分用に、ラスタースタートオーダーで読み込まれる現在のピクチャ用に1つずつのアドレスを発生させる。異なるセットアップは：インターレース／順送り型ディスプレイ、及び／もしくは垂直のアンサンプリング、及び（インターレースディスプレイに対する）フィールド同期化を可能にする。下位レベルでは、リードアドレス発生器はベースアドレスをブロックアドレス、及びDRAMのページ構造と互換性のある3つの成分の各々のためのバイトカウントのシーケンスに変換する。DRAMインターフェースに提供されるアドレスはブロックスタート及びブロックエンドカウントと共にページ及びラインアドレスである。

【1819】C. 1. 5 「出力パイプライン」

DRAMインターフェースからのデータは出力パイプラインに供給される。3つの成分ストリームはまず垂直に補間され、次に水平に補間される。補間の後、3つの成分は等しい割合（4：4：4）になっていなければならない、色空間変換器及び色ルックアップ表／ガンマ補正に送られる。出力インターフェースはディスプレイがHSYSCに達するまでこの時点でストリームを保持することができる。その後、出力コントローラは3つの成分を必要に応じて多重送信し、1つ、2つまたは3つの8ビットバスに向ける。

【1820】C. 1. 6 「タイミング様式」

基本的に、イメージフォーマット部と連合する2つの主要なタイミング様式がある。第1に、チップのフロントエンド（アドレス発生器及びバッファマネージャ、プラスDRAMインターフェースのフロントエンド）のためにタイミングを提供するシステムクロックがある。第2に、バックエンド（DRAMインターフェース出力、及び出力パイプライン全体）のために全てのタイミングを駆動させるピクセルクロックがある。

【1821】2つの前述のクロックの各々は多くのオンチップクロック発生器を駆動させる。FIFO、バッファマネージャ、及びリードアドレス発生器は同じクロック(DΦ)から操作し、ライトアドレス発生器は同様の、しかし別個のクロック(WΦ)を用いる。データは内部DRAMインターフェースクロック(outΦ)上のDRAMインターフェースにクロックされる。DΦ、WΦ、及びoutΦは全てsysclkから発生される。

【1822】リード及びライトアドレスはDRAMインターフェース自体のクロックによってDRAMインターフェースにクロックされる。

【1823】データはbifRΦ上のDRAMインターフェースから読み出され、NEΦによって表示されるクロック上で操作する、bushyne（その物理的ロケーションの故に、ノースイスト）と呼ばれる出力パイプラインのセクションに伝送される。前方のガンマRAMからのパイプラインセクションは別ではあるが類似したクロック(RΦ)上でクロックされる。bifRΦ、NEΦ、RΦは全てピクセルクロック、pixinから引き出される。

【1824】テストのため、ブロック間の主要なインターフェースの全ては付属のスノーバもしくはスーパースノーバの何れかを持っている。これはタイミング様式と必要なアクセスのタイプに依存する。別個ではあるが類似したタイミング様式間のブロック境界はそれに連合するリタイミングラッチを持っている。

【1825】セクションC. 2 「バッファ管理」

C. 2. 1 「序文」

本発明によるバッファ管理ブロックの目的は、ピクチャデータのライティング及びリーディング用に2つか3つの外部バッファを特定するインデックスをアドレス発生器に供給することである。これらのインデックスの割当ては、各々がオペレーションにおいてタイミング様式の1つの影響を表す3つの主要な要素により影響される。これらはピクチャデータがイメージフォーマット部への入力に到着する率（コード化データ率）、データが表示される率（表示データ率）、及びエンコード化ビデオシーケンスのフレーム率（提示率）である。

【1826】C. 2. 2 「機能的展望」

3バッファシステムは、システムのタイミング束縛があるとすれば、最善の可能なフレームシーケンスを達成するために、フレームが必要に応じて繰り返されるかスキップされるように、提示率及び表示率が異なる（例えば、2-3ブルダウン）ようにできる。ピクチャがデコードするため利用可能な表示時間より長くかかる場合、他の全てが「キャッチアップ」する間、前のフレームが繰り返されるように、デコーディングにおける困難さを表すピクチャを同じような方法で収容することができる。2バッファシステムでは、3つのタイミング様式がロックされなければならない—それはスラックを処理するためのフレキシビリティを提供する第3のバッファである。

【1827】バッファマネージャは各々の外部バッファと連合する特定のステータス情報を維持することにより操作する。これはバッファが使用中であるか、バッファにデータが一杯詰まっているか、あるいは表示の準備ができていないかを指示するフラグ、またバッファにおいて現在記憶されているピクチャシーケンス内のピクチャ数

を指示するフラグを含む。提示数も記録されるが、これはピクチャクロックパルスが受け取られる度に増分する数であり、エンコード化シーケンスのフレーム率に基づいて表示されることが現在予測されているピクチャ数を表す。

【1828】アライバルバッファ（入ってくるデータが書き込まれるバッファ）は、ピクチャスタートトークンが入力において検出される度に割り当てられる。このバッファにはその後IN USEのフラグが付けられる。ピクチャエンドに達すると、アライバルバッファは割り10
当てから外され（0にリセットされ）、ピクチャ数と提示数の間の関係に応じて、FULLもしくはREADYのフラグが付けられるバッファとなる。

【1829】ディスプレイアドレス発生器は、2線式インターフェースを介して、vsyncの度に一度、新しいディスプレイバッファをリクエストする。READYというフラグが付けられたバッファがあれば、それはバッファマネージャによって表示されるために割り当てられるであろう。READYバッファがなければ、前に表示されたバッファが繰り返される。

【1830】提示数が変化する度に、それは検出され、そのピクチャナンバーと提示数の間の関係を調べることで、完全なピクチャを含むあらゆるバッファのREADY-nessがテストされる。バッファは順番に考慮される。あるバッファがREADYであると思われる時、これは自動的に前にREADYというフラグが付けられていたバッファのREADY-nessをキャンセルする。その後、前のバッファにはEMPTYというフラグが付けられる。後に考慮されるバッファにおいて、20
割り当てによって後のピクチャ数が記憶されるので、こうしたことがうまく行く。

【1831】入力ストリーム内のスキップされたピクチャが指示されると、H. 261内の時間標準トークンがバッファのピクチャ数を修正させる。しかしながら、こうした特徴は、企図されてはいるが、現在のところ含まれてはいない。同様に、MPEG内の時間標準も何の影響も持たない。

【1832】フラッシュトークンは全てのバッファがEMPTYであるか、あるいはディスプレイバッファとして割り当てられるまで、入力をストールさせる。その後、提示数とピクチャ数がリセットされ、新しいシーケンスを始めることができる。

【1833】C. 2. 3 「アーキテクチャ」
C. 2. 3. 1 「インターフェース」
C. 2. 3. 1. 1 「bm frontに対するインターフェース」

全てのデータは入力FIFO、bm frontからバッファマネージャに入力される。この伝送は2線式インターフェースを介して行われ、データは8ビット幅プラス拡張ビットである。バッファマネージャに到着する全50

てのデータは完全なトークンであると保証される。これは提示数の連続プロセッシング、及びデータストリーム内に有効ギャップがある場合のディスプレイバッファリクエストのための必要条件である。

【1834】C. 2. 3. 1. 2 「waddrgenに対するインターフェース」

トークン（8ビットデータ、1ビット拡張）は2線式インターフェースを介してライトアドレス発生器に伝送される。ピクチャスタートトークンがwaddrgenに到着すると同時に、正しいインデックスがアドレス発生器のために利用できるように、アライバルバッファインデックスも同じインターフェース上で伝送される。

【1835】C. 2. 3. 1. 3 「dispaddrに対するインターフェース」

リードアドレス発生器へのインターフェースは、「リクエスト」と「アクノレッジ」信号として各々行動すると考えられる2つの別々の2線式インターフェースから成る。しかしながら、いずれかの端にある2つの2線式ベースのステートマシンの故に、単線式は適当ではない。

【1836】イベントのシーケンスは通常以下のようにdispaddrインターフェースと連合する。まず、dispaddrはバッファマネージャへのdrq valid入力を認定することにより、表示装置からvsyncに答えるリクエストを呼び出す。次に、バッファマネージャがそのステートマシン内の適切なポイントに達すると、バッファマネージャはリクエストをアクセプトし、表示されるバッファの割り当てに努める。その後、disp validワイヤが認定され、バッファインデックスが伝送されるが、これは典型的に、dispaddrによって直ちにアクセプトされる。更に、現在のインデックスと連合するフィールド数が前のフィールド数と無関係にリセットされなければならないことを指示する、この最後の2線式インターフェース（rst fld）と連合する追加ワイヤがある。

【1837】C. 2. 3. 1. 4 「マイクロプロセッサインターフェース」

バッファマネージャブロックは8ビットのデータバス及びリード/ライトストロブと共に、4ビットのマイクロプロセッサアドレススペースを使用する。2つのセレクト信号があり、1つはユーザーアクセス可能ローケーションを指示し、他方は通常の操作状態の下でアクセスを必要とすべきではないテストローケーションを指示する。

【1838】C. 2. 3. 1. 5 「イベント」

バッファマネージャは2つの異なるイベント、インデックスファウンドとレイトアライバルを作り出すことができる。この内最初のは、ピクチャが到着した時に認定され、そのピクチャスタート拡張バイト（ピクチャインデックス）はセリットアップ時にBU BM TARG

ET IXレジスタに書き込まれる値と適合する。第2のピクチャ数は現在の提示数より小さい、つまりバッファマネージャまでのシステムパイプラインにおけるプロセッシングは提示要件を満たして行くように処理してはいない。

【1839】C. 2. 3. 1. 6 「ピクチャクロック」

本発明では、ピクチャクロックは提示数カウンタに対するクロック信号であり、オンチップで発生されるか、あるいは外部ソース（通常はディスプレイシステム）から引き出される。バッファマネージャはこれらの信号の両方をアクセプトし、`pclk_ext`（バッファマネージャの制御レジスタ内のビット）の値に基づいて1つを選択する。この信号は更に、イメージフォーマット部がそれ自体のピクチャクロックを生じさせている場合、この信号もチップからの出力として利用できるように、パッド`picoutpad`用のイネーブルとして作用する。

【1840】C. 2. 3. 2 「主要なブロック」

以下のセクションはバッファマネージャ概略図（`bmlogic`）を作成する様々なハードウェアブロックについて説明する。

【1841】C. 2. 3. 2. 1 「入力/出力ブロック（`bminput`）」

このモジュールはバッファマネージャの4つの2線式インターフェース（入力データと出力データ、`drq valid/accept`及び`disp valid/accept`）と連合するハードウェアの全てを含む。入力データレジスタはそれに付属する幾つかのトークンデコーディングハードウェアと共に図示した。`bmto` 30 `kdec`に対する入力にある信号`vheader`は、ヘッダが有効であろう（つまり、トークンのまん中ではない）ポイントでのみトークンデコード出力が認定されることを確実にするために使用される。`rtimd`ブロックは、パイプライン内の次のブロック用の二重の入力データレジスタに隣接する出力データレジスタとして作用する。これは異なるクロック発生器故のタイミング差を説明する。信号`go`と`ngo`はデータ`valid`、`accept`及び`notstopped`のANDに基づいており、入力もしくは出力において何かが「大破」された 40 場合を指示するため、ステートマシーン内のどこかで使用される。このモジュールのディスプレイインデックス部分はデータに関する等価「ゴー」信号と共に、2線式インターフェースから成る。`rst fld`ビットもここで発生し、これは設定された場合、`disp valid`が1サイクルのために高くなるまで高いままである信号である。その後、それはリセットされる。それに加えて、フラッシュトークンがディスプレイバッファによって、全ての外部バッファに`EMPTY`または`IN USE`のフラグを付けさせた後、`rst fld`がリセッ 50

トされる。これはピクチャ数と提示数の両方がリセットされるのと同じポイントである。

【1842】階層の次のレベルに現れる入力データレジスタと連合する少量の追加回路がある。この回路は入力データレジスタが`BU BM TARGIX`に書き込まれるものと等しい値を持ち、それがイベント発生のために使用されることを指示する信号を作り出す。

【1843】C. 2. 3. 2. 2 「インデックスブロック（`bmindex`）」

インデックスブロックは主として、様々な戦略的バッファインデックスを表示する2ビットレジスタから成る。これらは`arr buf`、到着するピクチャデータが書き込まれるバッファ、`disp buf`、そこからピクチャデータが表示のために読み出されるバッファ、及び`rdy buf`、バッファが`dispaddr`によってリクエストされた場合、表示され得るほとんどの現在までのピクチャを含むバッファインデックスである。更に、`buf ix`を含むレジスタがあり、これはバッファに対する一般的なポインタとして使用される。このレジスタは増分され（マックスにD入力）、それらのステータスを調べるバッファを通して循環する、あるいはステータスを変更する必要がある場合、`arr buf`、`disp buf`、または`rdy buf`の1つの値が指定される。これらのレジスタ（`ph0`バージョン）の全ては、テストアドレススペースの一部として、マイクロプロセッサからアクセス可能である。`old ix`は`buf ix`のリタイムドバージョンであり、バッファステータス及び`im stus`ブロック内のピクチャ数レジスタを可能化するために使用される。`buf ix` 及び`old ix`は共にこのブロックから出力される3つの信号（各々が値1〜3を保持することができる）にデコードされる。他の出力は`buf ix`が`arr buf`もしくは`disp buf`のいずれかと同じ値を持っているかどうかを指示し、また`rdy buf`及び`disp buf`のいずれかが値0を持っているかどうかを指示する。0はバッファに対する参照符号ではない。それは単に現在割り当てられる`arrival/display/ready`バッファがないことを指示するだけである。

【1844】`arr buf`及び`disp buf`は各々の2線式インターフェース出力アクセプトレジスタによって可能化される。

【1845】`bmlogic`レベルの追加回路は現在のバッファインデックス（`buf ix`）がセットアップ時に制御レジスタに書き込まれる値によって限定されるように、使用中の最大インデックスに等しいか否かを判断するために使用される。制御レジスタ内の「1」は3バッファシステムを指示し、「0」は2バッファシステムを指示する。

【1846】C. 2. 3. 2. 3 「バッファステータ

ス」
バッファステータス内の主要な成分はステータスと各々のバッファ用のピクチャ数レジスタである。3つのグループの各々はマスター・スレーブ配置であり、その場合、スレーブは3つのレジスタのバンクであり、マスターはその出力が（レジスタ enables を使用して、old ix によって切り換えられる）スレーブの1つに向けられる1つのレジスタである。マスターに対する可能な入力1つは（bm logic レベルで buf ix によってインデックスされる）異なるスレーブ出力の間で多重送信される。bm logic レベルでデコードされるバッファステータスはステートマシンロジックにおいて使用されるため、表237に示された値のいずれをも取ることができ、あるいはその前の値を再循環させることができる。ピクチャ数は前の値または1だけ（または1+デルタ、H. 261の場合に実際のものと予測される時間リファレンス間の差）増分された前の値を取ることができる。この値はブロックに存在する8ビットアダーによって供給される。このアダーに対する最初の入力は this pnum、現在書き込まれているデータのピクチャ数である。

【1847】

【表237】

バッファステータス	バリュー
EMPTY	00
FULL	01
READY	10
IN USE	11

表C. 2. 1 バッファステータスバリュー

3 バッファピクチャ数レジスタのいずれも、（殆ど常に時代遅れになっている）それら自身の前のピクチャ数ではなく、現在（または前の）ピクチャ数に基づいて容易に更新できるように、これは別個に（それ自身のマスター・スレーブ配置に）記憶される必要がある。this pnum は-1 にリセットされ、最初のピクチャが到着する時、アダーから出力に加算されるので、最初のバッファピクチャ数レジスタへの入力は0である。

【1848】現在のバージョンにおいては、その値を供給すべき時間リファレンスブロックの欠如により、デルタは0に接続される。

【1849】C. 2. 3. 2. 4 「提示数」

8ビット提示数レジスタは、提示数が最後に調べられてから変化していることを指示するため、ステートマシンにおいて使用される関連提示フラグを持つ。こうしたことが必要であるのは、ピクチャクロックが本質的に同

期的であり、如何なるステートにおいても活性であり得るからであり、これらが提示数に関係するからではない。本ブロック内の回路の残りはピクチャクロックパルスが発生したことを検出し、この事実を「覚えておく」ことに関係する。この方法で、提示数は更新することが有効である時に更新することができる。イベントの代表的シーケンスを図165に示す。信号 incr prn はリタイムドピクチャクロックライジングエッジの1サイクル後に活性となり、ステートが入ってくるまで持続し、その間に提示数を修正することができる。これは信号 en prnum によって指示される。特定のステートの間にのみ提示数の更新ができるようにした理由は、それが信号 rdyst を提供するための standard-cell、not-very-fast 8ビットアダーを含むロジックのかんりの量を駆動させるために使用されるからである。従って、それは次に続くステートがその結果を使用していないステートの間にのみ変更されなければならない。

【1850】C. 2. 3. 2. 5 「時間リファレンス」

本発明による時間リファレンスブロックは、イメージフォーマッティング部の現在の態様からは省略されているが、仕上げるためにそのオペレーションについてここで説明する。

【1851】本ブロックの機能はデルタ、H. 261 データストリーム内のトークンにおいて受け取った時間リファレンスバリューと、「予測された」時間リファレンス（1+前のバリュー）の間の差を計算することである。これによりH. 261においてフレームをスキップすることができる。時間リファレンストークンは全ての non-H. 261 ストリームにおいて無視される。計算されたバリューはバッファ用のピクチャ数を計算するためにステータスブロックにおいて使用される。bm logic からのブロックを省略する効果は、H. 261 ストリームが幾つかのピクチャ数をスキップすべきであると指示しても、ピクチャ数がどのシーケンスにおいても常に連続していることである。

【1852】（概略図 bm tref において見ることができる）ブロックの主要な成分は tr、exp tr 及びデルタ用のレジスタである。発明においては、tr は0にリセットされ、適当な場合に入力データレジスタからロードされるものである。同様に、exp tr は-1にリセットされ、時間リファレンスステートのシーケンスの間に、1またはデルタだけ増分される。それに加えて、デルタは0にリセットされ、他の2つのレジスタの間の差がロードされる。3つの全てのレジスタはフラッシュトークンの後リセットされる。このブロック内のアダーはデルタと exp tr の両方の計算のために、つまり減算及び加算操作のために各々使用され、信号 del t a c a l e によって制御される。

【1853】C. 2. 3. 2. 6 「制御レジスタ (bm uregs)」

バッファマネージャ用の制御レジスタはブロックbm uregsにある。これらはアクセスビットレジスタ、(外部バッファの最大数及び内部/外部ピクチャクロックを限定する) セットアップレジスタ、及びターゲットインデックスレジスタである。アクセスビットは予測されたように同期化される。信号stopd 0、stopd 1、及びnstopd 1はアクセスビットのOR及び2個のイベントストップビットから引き出される。全てのbmlogic用のUpiアドレスデコーディングはブロックbm udecによって行われ、それはイメージフォーマッティング部のトップレベルのアドレスデコードからの2つのセレクト信号と共に、upiデータバスの下位4ビットを取る。

【1854】C. 2. 3. 2. 7 「制御用ステートマシン」

ステートマシンロジックは元々それ自体のブロック、bm stateを占有していた。しかしながら、コー

ド発生理由のため、今では平板化され、bmlogic概略図のシート2の上にある。

【1855】このロジックの主要セクションは同じである。これはデコーディング、他のbmlogicブロックの制御用のロジック信号の発生、及びステートマシンを通るルートを選択するために使用されるフラグ、fromps及びfromflを含む新しいステートエンコーディングを含む。bmstus及びbmindex用にマックス制御信号を作るための別個のブロックがある。

【1856】ステートマシンハードウェアにおける信号には、タイピング及び参照のしやすさのために簡単なアルファベット名を与えてきた。それらの全てを、それらが表すロジック表現と共に表238～表241にリストアップする。更にそれらはbmlogicの行動M. 解説 (bmlogic. M) におけるコメントとしても現れている。

【1857】

【表238】

信号名	ロジック表現
A	ST PRES1, presflg, (bstate==FULL), rdyst, (rdy==0), (ix==max)
B	ST PRES1, presflg, (bstate==FULL), rdyst, (rdy==0), (ix!=max)
C	ST PRES1, presflg, (bstate==FULL), rdyst, (rdy!=0),
D	ST PRES1, presflg, !((bstate==FULL), rdyst), (ix==max)
E	ST PRES1, presflg, !((bstate==FULL), rdyst), (ix!=max)
F	ST PRES1, presflg
G	ST DRQ, drq valid, disp acc, (rdy==0), (disp!=0)
PP	ST DRQ, drq valid, disp acc, (rdy==0), (disp!=0), fromps
QQ	ST DRQ, drq valid, disp acc, (rdy==0), (disp!=0), fromfl
RR	ST DRQ, drq valid, disp acc, (rdy==0), (disp!=0), !(fromps+fromfl)

表C. 2. 2 ステートマシンにおいて使用される信号名 (1/4)

【1858】

【表239】

信号名	ロジック表現
H	ST DRQ, drq valid, disp acc, (rdy!=0), (disp!=0)
I	ST DRQ, drq valid, disp acc, (rdy!=0), (disp==0)
J	ST DRQ, drq valid, disp acc, (rdy==0), (disp==0), fromps
NN	ST DRQ, drq valid, disp acc, (rdy==0), (disp==0), fromfl
OO	ST DRQ, drq valid, disp acc, (rdy==0), (disp==0), !(fromps+fromfl)
K	ST DRQ, !(drq valid, disp acc), fromps
LL	ST DRQ, !(drq valid, disp acc), fromfl
MM	ST DRQ, !(drq valid, disp acc), !(fromps+fromfl)
L	ST TOKEN, ivr, oar, (idr==TEMPORAL REFERENCE)
SS	ST TOKEN, ivr, oar, (idr==TEMPORAL REFERENCE), H. 261

表C. 2. 2 ステートマシンにおいて使用される信号名 (2/4)

【1859】

【表240】

信号名	ロジック表現
TT	ST TOKEN, ivr, oar, (idr==TEMPORAL REFERENCE), H. 261
M	ST TOKEN, ivr, oar, (idr==FLUSH)
N	ST TOKEN, ivr, oar, (idr==PICTURE START)
O	ST TOKEN, ivr, oar, (idr==PICTURE END)
P	ST TOKEN, ivr, oar, (idr==<OTHER TOKEN>)
JJ	ST TOKEN, ivr, oar, (idr==<OTHER TOKEN>), in extn
KK	ST TOKEN, ivr, oar, (idr==<OTHER TOKEN>), ! in extn
Q	ST TOKEN, ivr, oar
S	ST PICTURE END, (ix==arr), ! rdystoar
T	ST PICTURE END, (ix==arr), rdyst, (rdy==0), oar
U	ST PICTURE END, (ix==arr), rdyst, (rdy!=0), oar
VV	ST PICTURE END, !oar
RorVV	ST PICTURE END, ! ((ix==arr), oar)
V	ST TEMP REF0, ivr, oar

表C. 2. 2 ステートマシンにおいて使用される信号名 (3/4)

[1860]

[表241]

信号名	ロジック表現
W	ST TEMP REF0, ! (ivr, oar)
X	ST OUTPUT TAIL, ivr, oar
FF	ST OUTPUT TAIL, ivr, oar, ! in extn
Y	ST OUTPUT TAIL, ! (ivr, oar)
GG	ST OUTPUT TAIL, ! (ivr, oar), in extn
DD	ST FLUSH, (ix==max), ((bstate==VAC) + ((bstate==USE), (ix==disp)))
Z	ST FLUSH, (ix!=max), ((bstate==VAC) + ((bstate==USE), (ix==disp)))
DDorEE	! ((bstate==VAC) + ((bstate==USE), (ix==disp)) + (ix==max))
AA	ST ALLOC, (bstate==VAC), oar
BB	ST ALLOC, (bstate==VAC), (ix==max)
CC	ST ALLOC, (bstate==VAC), (ix!=max)
UU	ST ALLOC, !oar

表C. 2. 2 ステートマシンにおいて使用される信号名 (4/4)

C. 2. 3. 2. 8 「モニタリングオペレーション」 50 (bmlhfb)

本発明では、バッファステータス情報、インデックスバリュー、及び提示数をシミュレーション中に観察できるように、モジュール、bminfoが含まれる。それはMの中に書き込まれ、その入力の1つが変化する度に出力を生じさせる。

【1861】C. 2. 3. 3 「レジスタアドレスマップ」

バッファマネージャのアドレススペースは2つのエリア

ア、ユーザー・アクセシブルとテストにスプリットされる。従って、トップレベルにおけるレンジデコードから引き出される2つの別個のイネーブルワイヤがある。表242はユーザー・アクセシブルレジスタを示し、表243はテストスペースの内容を示す。

【1862】

【表242】

レジスタ名	アクセス	バス	リセット ステート	機能
BU BM ACCESS	0x10	[0]	1	バッファマネージャ用アクセスビット
BU BM CTL0	0x11	[0]	1	最大バッファ: 1→3 バッファ: 0→2
		[1]	1	外部ピクチュア クロックセレクト
BU BM TARGET IX	0x12	[3:0]	0x0	ピクチュアの到着 を検出するため
BU BM PRESS NUM	0x13	[7:0]	0x00	提示数
BU BM THIS PNUM	0x14	[7:0]	0xFF	現在のピクチュア 数
BU BM PIC NUM0	0x15	[7:0]	none	バッファ1内の ピクチュア数
BU BM PIC NUM1	0x16	[7:0]	none	バッファ2内の ピクチュア数
BU BM PIC NUM2	0x17	[7:0]	none	バッファ3内の ピクチュア数
BU BM TEMP REF	0x18	[4:0]	0x00	ストリームからの 時間リファレンス

表C. 2. 3 ユーザー・アクセシブルレジスタ

【1863】

【表243】

レジスタ名	アクセス	バス	リセット ステート	機能
BU BM PRES FLAG	0x80	[0]	0	提示フラグ
BU BM EXP TR	0x81	[4:0]	0xFF	予想時間リファレ ンス
BU BM TR DELTA	0x82	[4:0]	0x00	デルタ
BU BM ARR IX	0x83	[1:0]	0x0	アライバルバッ ファインデックス
BU BM DSP IX	0x84	[1:0]	0x0	ディスプレイバッ ファインデックス
BU BM RDY IX	0x85	[1:0]	0x0	レディバッファ インデックス
BU BM BSTATE3	0x86	[1:0]	0x0	バッファ3ステー タス
BU BM BSTATE2	0x87	[1:0]	0x0	バッファ2ステー タス
BU BM BSTATE1	0x88	[1:0]	0x0	バッファ1ステー タス
BU BM INDEX	0x89	[1:0]	0x0	現在のバッファ インデックス
BU BM STATE	0x8A	[4:0]	0x00	バッファマネー ジャステート
BU BM FROMPS	0x8B	[0]	0x0	PICTURE STARTフラグ から
BU BM FROMFL	0x8C	[0]	0x0	FLUSH TOKENフラグ から

表C. 2. 4 テストレジスタ

C. 2. 4 「ステートマシンのオペレーション」

表244に詳述するように、バッファマネージャのステートマシンには19のステートがある。これらは図1 30 66に示すように、また行動解説bmlogic.Mにも記載するように相互作用する。

【1864】

【表244】

ステート	バリュー
PRES0	0x00
PRES1	0x10
ERROR	0x1F
TEMP REF0	0x04
TEMP REF1	0x05
TEMP REF2	0x06
TEMP REF3	0x07
ALLOC	0x03
NEW EXP TR	0x0D
SET ARR IX	0x0E
NEW PIC NUM	0x0F
FLUSH	0x01
DRQ	0x0B
TOKEN	0x0C
OUTPUT TAIL	0x08
VACATE RDY	0x17
USE RDY	0x0A
VACATE DISP	0x09
PICTURE END	0x02

表C. 2. 5 バッファステータス

C. 2. 4. 1 「リセットステート」

リセットステートはPRES0であり、主ループが初期
50 に循環されるように、フラグは0に設定される。

557

【1865】C. 2. 4. 2 「主ループ」

ステートマシンの主ループが図167に示すようなステート（主な線図-図166において強調している）から成る。ステートPRES0とPRES1は信号pres_flgを介してピクチャクロックを検出することに関係する。2個のサイクルはそれらが全てrdyst、C. 2. 3. 2. 4において説明したアダー出力信号のバリュに格納するので、ピクチャクロックを含めることができる。提示フラグが検出された場合、全てのバッファは可能な「レディネス」のために調べられ、そうでなければ

```
(pic num>pres num)
  &&(pic num-pres num)>=128)
or
  (pic num<pres num)
    &&(pres num-pic num)<=128)
or
  pic num==pres num
```

ステートDRQはディスプレイバッファ用のリクエスト(drq_validreg && disp_acc reg)をチェックする。もし如何なるリクエストもなければ、ステートは（通常は、後述するように、ステートTOKENに）進む。そうでなければ、ディスプレイバッファインデックスが以下のように発せられる。レディバッファがなければ、前のインデックスが再発行され、あるいは、前のディスプレイバッファがなければ、ゼロインデックス（ゼロ）が発せられる。バッファが表示される準備が整っていれば、そのインデックスが出され、そのステートが更新される。必要であれば、前のディスプレイバッファがクリアされる。ステートマシンはその後以前のように前進する。

【1866】ステートTOKENは主ループを完了させるための典型的なオプションである。有効な入力があり、出力がストールされていなければ、トークンは（後のセクションで説明する）戦略的バリュのために調べられ、そうでなければ、制御はステートPRES0に戻る。

【1867】制御はある条件が満たされる時に、主ループから分岐するだけである。これらについては以下のセクションにおいて説明する。

【1868】C. 2. 4. 3 「レディバッファインデックスの割当」

PRES0-PRES1ループの間に、バッファがレディであると判断されれば、前のレディバッファは空にされる。なぜなら、いつでも1つだけのバッファがレディと指定され得るからである。ステートVACATE_RDYはそのステートをVACANTに設定することにより、古いレディバッファをクリアし、制御がPRES0ステートに戻る時、全てのバッファのレディネスがテストされるように、バッファインデックスを1にリセットする。こうすることの理由は、インデックスが今では前

558

ば、ステートマシンはステートDRQに進む。PRES0-PRES1ループの回りの各サイクルは異なるバッファを調べ、フル及びレディ条件をチェックする。これらが満たされると、前のレディバッファ（もし存在していれば）がクリアされ、新しいレディバッファが割り当てられ、そのステータスが更新される。このプロセスは全てのバッファが調べられる（インデックス==max_index）まで繰り返され、それからステートが前進する。以下のいずれかが満たされれば、バッファは表示される準備が整ったと考えられる：

のレディバッファを指しており（それをクリアする目的）、我々の意図する新しいレディバッファインデックスの記録がないからである。従って、全てのバッファを再テストすることが必要である。

【1869】C. 2. 4. 4 「ディスプレイバッファインデックスの割当」

ディスプレイバッファインデックスの割当はステートDRQ（ステートUSERDY）から直接、あるいは古いディスプレイバッファステートをクリアするステートVACATE_DISPを介して行われる。選ばれたディスプレイバッファはIN_USEというフラグが付けられ、rdy_bufのバリュが0に設定され、ステートDRQに戻るためにインデックスは1にリセットされる。更に、disp_bufに必要なインデックスが与えられ、2線式インターフェースワイヤ(disp_validとdrq_acc)がそれに依じて制御される。ステートTOKEN、FLUSH、ALLOC間の決定がステートUSERDYにおいて為される必要がないようにするためだけに、制御はステートDRQに戻る。

【1870】C. 2. 4. 5 「ピクチャエンドを受け取った時のオペレーション」

ピクチャエンドトークンを受け取ると、制御はステートTOKENからステートピクチャエンドに移り、そこで、インデックスがまだ現在のアライバルバッファを指していない場合、そこを指すように設定され、そのステートが更新され得る。out_acc_reg及びen_fullが当てはまると仮定すれば、ステータスを後述するように更新することができる。そうでなければ、制御はそれらが共に当てはまるまでステートピクチャエンドに残る。en_full信号がライトアドレス発生器によって供給され、スイングバッファがスイングしたことを、つまり最後のブロックがうまく書き込まれ、従

ってバッファステータスを更新しても安全であることを指示する。

【1871】完了したばかりのバッファのレディネスがテストされ、テストの結果に応じてステータスFULLまたはREADYのいずれかが与えられる。それがレディであれば、rdy bufにそのインデックスのバリューが与えられ、set la ev信号（レイアウトイベント）が高く設定される（予測されるディスプレイがデコーディングの適時に前に進んだことを指示する）。arr bufの新しいバリューは今は0であり、前のレディバッファがそのステータスをクリアする必要があれば、インデックスはそこを指すように設定され、制御はステートVACATE RDYに移動する。そうでなければ、インデックスは1にリセットされ、制御は主ループのスタートに戻る。

【1872】C. 2. 4. 6 「ピクチャスタートを受け取った時のオペレーション（アライバルバッファの割当）」

ピクチャスタートトークンがステートTOKENの間に到着すると、フラグfrom psが設定され、ステートTOKENの代わりにステートALLOCを巡視するように、ベーシックステートマシンループを変更させる。ステートALLOCはアライバルバッファ（その中に到着するピクチャデータを書き込むことができる）の割当に関係し、そのステータスがVACANTであるものを見つけるまでバッファを通して循環する。バッファはデータ2線式インターフェースの上に出力されるので、out acc regが高い場合にのみ割り当てられる。従って、ループのまわりの循環はこれが本当にそうしたことになるまで続くであろう。適当なアライバルバッファが見つけれられ、インデックスはarr bufに割り当てられ、そのステータスにはIN USEというフラグが付けられる。インデックスは1に設定され、フラグfrom psがリセットされ、ステートはNEW EXP TRに進む用に設定される。（ピクチャスタートに続くワードに含まれる）ピクチャのインデックスがチェックされ、それがtarg ix（セットアップで指定されたターゲットインデックス）と同じであるか否かが判断され、もしそうであれば、set i f+ ev（インデックスファウンドイベント）が高く設定される。

【1873】3つのステータスNEW EXP TR、SET ARR IX、及びNEWPIC NUMが新しく予測される時間リファレンス及び入ってくるデータ用のピクチャ数を設定する。正しいピクチャ数レジスタが更新される（this pnumも更新されることに注意）ように、中間ステートはインデックスをarr bufに設定する。その後制御は、低い拡張に遭遇するまで、（好ましい2線式インターフェース信号を仮定して）データを出力するステートOUTPUTTAILに

進む。この時点で、主ループは再スタートされる。これは全データブロック（64アイテム）が出力されることを意味し、その間に提示フラグもしくはディスプレイリクエスト用のテストは行われぬ。

【1874】C. 2. 4. 7 「フラッシュを受け取った時の動作」

データストリーム内のフラッシュトークンはシーケンス情報（提示数、ピクチャ数、rst fld）をリセットすべきことを指示する。これはFLUSHに導く全てのデータが正しく処理された時にのみ発生することができる。従って、FLUSHを受け取った後、全てのフレームがディスプレイに手渡された、つまり、1つのバッファを除いて全てのバッファがステータスEMPTYを持ち、その1つのバッファが（ディスプレイバッファとして）IN USEであることが確実になるまで、バッファの全てのステータスを監視することが必要である。その時点で、「新しいシーケンス」を安全に使用することができる。

【1875】フラッシュトークンがステートトークンにおいて検出されると、フラグfrom flが設定され、ステートトークンの代わりにステートフラッシュを巡視するように、ベーシックステートマシンループを変更させる。ステートFLUSHは各バッファのステータスを順に調べ、それがVACANTもしくはディスプレイとしてIN USEになるのを待つ。ステートマシンは条件が当てはまるまで、単にループのまわりを循環し、それからそのインデックスを増分して、全てのバッファを訪問してしまうまでプロセスを繰り返す。最後のバッファが条件を満たすと、提示数、ピクチャ数、及び全ての時間リファレンスレジスタがそれらのリセット値rst fldが1に設定されると仮定する。フラグfrom flがリセットされ、通常的主ループオペレーションが再び始められる。

【1876】C. 2. 4. 8 「時間標準を受け取った時の動作」

時間標準トークンに遭遇すると、H. 261ビットのチェックが行われ、設定されると、4つのステートTEMP REF0~TEMP REF3を巡視する。これらは次のオペレーションを遂行する。

【1877】TEMP REF0: temp ref = in data reg;
TEMP REF1: delta = temp ref - exp tr; index = arr buf;
TEMP REF2: exp tr = delta + exp tr;
TEMP REF3: pic num[i] = this pnum + delta; index = 1.

C. 2. 4. 9 「他のトークンとテール」

ステートトークンは上記において概略したもの以外の全ての場合において、ステートOUTPUTTAILに

制御を送る。最後のトークンワードに遭遇する (in extn regが低い) まで制御はここに留まり、主ループはそれから再突入される。

【1878】C. 2. 5 「アプリケーションノート」
C. 2. 5. 1 「バッファマネージャ入力をストールするステートマシーン」

この要件はピクチャクロックとディスプレイバッファリクエストの「H. 261」タイミングイベントを繰り返してチェックする。これらのチェックの間に、バッファマネージャ入力をストールさせることの必要性は、バッファマネージャの入力に連続的なデータ供給がある時、バッファマネージャを通るデータ率に制限があることを意味する。典型的なステートシーケンスはPRES0、PRES1、DRQ、TOKEN、OUTPUT TAILであり、OUTPUT TAILを除き、各々が1サイクル続く。これは各64データアイテムのブロックのために、入力がストールされている間 (ステートPRES0、PRES1、DRQの間) 3サイクルのオーバーヘッドがあり、それによって3/64もしくはほぼ5%だけライト速度を遅くさせることを意味する。このオーバーヘッドの数は、ステートマシンの補助ブランチが最悪の場合の条件下で実行される時、時には13サイクルまで増加させることができる。このように多くのオーバーヘッドはフレーム毎に一度のベースで適用できるだけであることに注意。

【1879】C. 2. 5. 2 「アクセス中の提示数行為」

C. 2. 3. 2. 4に示す概略図により図示されるbm presの特別な態様は、提示数がupiアクセス中に自由に動くことを意味する。アクセスを廃棄する時の提示数がアクセスを得た時の提示数と同じであることが求められる場合、これはアクセスが認められた後の提示数を読み、それをアクセスが廃棄される直前に書き戻すことによって成し遂げられる。注意すべきことは、これは非同期的であり、従って更に効果性を確かめるために、そのアクセスを数回繰り返すことが望ましいことである。

【1880】C. 2. 5. 3 「H261時間リファレンス数」

モジュールbm tref (図示せず) はbmlogi cの中に含まれるべきである。H. 261時間リファレンスバリューは、bm trefからのデルタ入力をbm stusモジュールに向けさせることにより正しく処理される。デルタ入力は、フレームが常に連続する場合に0に結び付けることができる。

【1881】セクションC. 3 「ライトアドレス発生」

C. 3. 1 「序文」

本発明によるライトアドレス発生ハードウェアの機能は、バッファに書き込むべきデータ用のブロックアドレ

スを作り出すことである。これはバッファのベースアドレス、ストリーム内で指示される成分、マクロブロック内の水平と垂直のサンプリング、ピクチャディメンション及びコーディングスタンダードを考慮する。データはマクロブロックの形態で到着するが、ディスプレイのためにラインを容易に検索できるように記憶されなければならない。

【1882】C. 3. 「機能的展望」

新しいブロックが (データトークンが指示する) テータストリームに到着する度に、ライトアドレス発生器は新しいブロックアドレスを作るように要請される。アドレスが実際に必要になる前に、64までのデータワードを (スイングバッファ内の) DRAMインターフェースによって記憶することができるので、直ちにアドレスを作ることが必要ではない。これは様々なアドレス成分を連続サイクル内の現在の合計に加算することができ、従って如何なるハードウェア乗算器のための必要事項をも除去することができることを意味する。マクロブロックカウンタ機能は戦略的終末値を記憶し、各ブロックアドレス計算の後の比較及び条件付き更新用のオペランドである、レジスタファイル内のカウントを動かすことによって達成される。

【1883】図170に示したピクチャフォーマットを考えてみると、予測されるアドレスシーケンスは標準のデータストリーム及びH. 261のようなデータストリームの双方から引き出すことができる。これらを下記に示す。スライスが充分な幅を持たない (11よりむしろ3マクロブロックである) が、ここでは便利さのために同じ「半ピクチャ幅スライス」コンセプトを使用し、シーケンスは「H. 261タイプ」のものであると仮定するので、フォーマットは実際にはH. 261規定に準じてはいないことに注意。データはフルマクロブロック、図示した例では4:2:0として到着し、各成分は指定されたバッファのそれ自身のエリアに記憶される。

標準アドレスシーケンス:

000, 001, 00C, 00D, 100, 200;
002, 003, 00E, 00F, 101, 201;
004, 005, 010, 011, 102, 202;
006, 007, 012, 013, 103, 203;
008, 009, 014, 015, 104, 105;
00A, 00B, 016, 017, 105, 205;
018, 019, 024, 025, 106, 107;
01A, 01B, 026,

080, 081, 08C, 08D, 122, 222;
082, 083, 08E, 08F, 123, 223;

H. 261型シーケンス:

000, 001, 00C, 00D, 100, 200;
002, 003, 00E, 00F, 101, 201;
004, 005, 010, 011, 102, 202;

018, 019, 024, 025, 106, 107;
 01A, 01B, 026, 027, 107, 207;
 01C, 01D, 028, 029, 108, 208;
 030, 031, 03C, 03D, 10C, 20C;
 032, 033, 03E, 03F, 10D, 20D;
 034, 035, 040, 041, 10E, 20E;
 006, 007, 012, 013, 103, 203;
 008, 009, 014, 015, 104, 105;
 00A, 00B, 016, 017, 105, 205;
 01E, 01F, 02A, 02B, 109, 209;
 020, 021, 02C, 02D, 10A, 20A;
 022, 023, 02E, 02F, 10B, 20B;
 036, 037, 042, 043, 10F, 20F;
 038, 039, 044, 045, 110, 210;
 03A, 03B, 046, 047, 111, 211;
 048, 049, 054, 055, 112, 212;
 04A, 04B, 056,

.
 06A, 06B, 076, 077, 11D, 21D;
 07E, 07F, 08A, 08B, 121, 221;
 080, 081, 08C, 08D, 122, 222;
 082, 083, 08E, 08F, 123, 223;
 C. 3. 3 「アーキテクチャ」
 C. 3. 3. 1 「インターフェース」
 C. 3. 3. 1. 1 「バッファマネージャに対するインターフェース」

バッファマネージャはデータ及びバッファインデックスを直接ライトアドレス発生器に出力する。これは2線式インターフェースの制御下に行われる。ある方法では、ライトアドレス発生器ブロックをバッファマネージャの拡張と考えることができる。なぜなら、その両者が非常に密接につながれるからである。しかしながら、それらは2つの別個の(しかし類似した)クロック発生器から操作する。

【1884】C. 3. 3. 1. 2 「dramifに対するインターフェース」

ライトアドレス発生器はDRAMインターフェース用のデータとアドレスを提供する。これらの各々はそれら自身の2線式インターフェースを持ち、dramifは異なるクロック様式においてその各々を使用する。特に、アドレスはライトアドレス発生器クロックに関連しないクロック上でdramifにクロックされる。従って、それは出力において同期化される。

【1885】C. 3. 3. 1. 3 「マイクロプロセッサインターフェース」

ライトアドレス発生器は8ビットのデータバス及びリード/ライトストロブと共に、3ビットのマイクロプロセッサアドレススペースを使用する。レジスタアクセス用に1つのセレクトビットがある。

【1886】C. 3. 3. 1. 4 「イベント」

ライトアドレス発生器は5個の異なるイベントを作り出すことができる。2個はデータストリーム(hmbsとvmbs)に現れるピクチャサイズ情報に答え、3個はDEFINE SAMPLINGトークン(各成分のために1イベント)に答える。

【1887】C. 3. 3. 2 「基本構造」

ライトアドレス発生器の構造を概略図waddrgen.schに示す。それはデータバス、幾つかの制御ロジック、及びスノーバと同期化から成る。

10 【1888】C. 3. 3. 2. 1 「データバス(bwadpath)」

データバスは本文書のC. 5章において説明したタイプのものであり、18ビットのアダー/サブトラクター、及びレジスタファイル(C. 3. 3. 4を参照)から成り、制御ロジックにおいて使用するため(アダーの出力に基づく)ゼロフラグを作り出す。

【1889】C. 3. 3. 2. 2 「制御ロジック」

本発明の制御ロジックはレジスタファイルロード及びドライバ信号、アダー制御信号、2線式インターフェース信号の全てを発生させるため、ハードウェアで構成され、更に書き込み可能制御レジスタを含む。

【1890】C. 3. 3. 2. 3 「スノーバ及び同期化」

スーパースノーバがデータとアドレスポートの両方に存在する。データバス内のスノーバはzcelsからのスノーバとして制御される。アドレスはライトアドレス発生器クロックとdramifのclk様式との間で同期化を持つ。Syncifsは2線式インターフェース信号用のzcelsにおいて使用され、簡略化されたシンクロナイザーがアドレス用のデータバスにおいて使用される。

【1891】C. 3. 3. 3 「制御ロジック及びステートマシン」

C. 3. 3. 3. 1 「入力/出力ブロック(wainout)」

このブロックは(トークンデコード用の)入力データ用ラッチ及び(4つの方法でデコーディングするための)アライバルバッファインデックスと共に、入力と2つの出力の2線式インターフェースを含む。

40 【1892】C. 3. 3. 3. 2 「2つのサイクル制御ブロック(wafc)」

フラグfc(最初のサイクル)がここに維持され、ステートマシンが2サイクルオペレーション(つまり、加算を含むオペレーション)の中間にあるかどうかを指示する。

【1893】C. 3. 3. 3. 3 「成分カウント(wacomp)」

別のアドレスが各成分内のデータブロックのために必要とされ、このブロックは入力ストリームにおいて受け取られるデータヘッダのタイプに基づいて考慮中の現在の

成分を維持する。

【1894】C. 3. 3. 3. 4 「モジュロ-3制御 (wa mod3)」

H. 261データストリーム用のアドレスシーケンスを発生させる時、スクリーンに沿って半分までマクロブロックの3つのローをカウントする必要がある (C. 3. 2を参照)。これはマクロブロックの新しいローが巡視される度に増分されるモジュロ-3カウンタを維持することによって達成される。

【1895】C. 3. 3. 3. 5 「制御レジスタ (wa uregs)」

モジュールwa uregsはステップアップレジスタ及びコーディングスタンダードレジスタを含み、後者はデータストリームからロードされる。セットアップレジスタは3ビット: QCIF (1sb) と、データストリーム内で予測される最大量の成分 (ビット1及び2) を使用する。更に、アクセスビットはこのブロックにあり (いつものように同期化され)、「ストップ」ビットはアクセスビット及びイベントストップビットのORとして、階層 (walogic) の次の上のレベルにおいて引き出される。マイクロプロセッサアドレスデコーディングは、リード及びライトストロブ、セレクトワイヤ、及びアドレスバスの低い方の2ビットを取るブロックwa udecによって行われる。

【1896】C. 3. 3. 3. 6 「制御ステートマシ

ン (wa state)」

このブロック内のロジックは幾つかの別個のエリアに分けられる。ステートデコード、新しいステートエンコード、「中間」ロジック信号の起源、データバス制御信号 (drivea, driveb, load、アダー制御及びセレクト信号)、乗算器制御、2線式インターフェース制御、及び5つのイベント信号である。

【1897】C. 3. 3. 3. 7 「イベント発生」

5つのイベントビットは入力に達する特定のトークンの結果として発生される。重要なことは、各々の場合に、イベントサービスルーチンが新しく受け取ったバリューに基づいて計算を行うので、イベントが発生される前に全トークンが受け取られることである。このため、各々のビットはイベントハードウェアに入力される前に全サイクルによって遅延される。

【1898】C. 3. 3. 4 「レジスタアドレスマップ」

ライトアドレス発生器ブロックには2セットのレジスタがある。これらは標準のセルセクションに置かれるトップレベルのセットアップタイプのレジスタであり、キーホールデータバスレジスタである。これらを表245及び表246～表256に各々リストアップする。

【1899】

【表245】

レジスタ名	アドレス	ビット	リセットステート	機能
BU WADDR COD STD	0x4	2	0	データストリームからのコーディングスタンダード
BU WADDR ACCESS	0x5	1	0	アクセスビット
BU WADDR CTL1	0x6	3	0	最大量の成分 [2:1]及び QCIF [0]
BU WA ADDR SNP2	0xB0	8		ライトアドレス発生器 アドレスc/p
BU WA ADDR SNP1	0xB1	8		
BU WA ADDR SNP0	0xB2	8		
BU WA DATA SNP1	0xB4	8		WAのデータ出力上の スナッチ
BU WA DATA SNP0	0xB5	8		

表C. 3. 1 トップレベルレジスタ

【1900】

【表246】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR BUFFER0 BASE MSB	0x85	2	ロードされ なければならない
BU WADDR BUFFER0 BASE MID	0x86	8	
BU WADDR BUFFER0 BASE LSB	0x87	8	
BU WADDR BUFFER1 BASE MSB	0x89	2	ロードされ なければならない
BU WADDR BUFFER1 BASE MID	0x8a	8	
BU WADDR BUFFER1 BASE LSB	0x8b	8	
BU WADDR BUFFER2 BASE MSB	0x8d	2	ロードされ なければならない
BU WADDR BUFFER2 BASE MID	0x8e	8	
BU WADDR BUFFER2 BASE LSB	0x8f	8	
BU WADDR COMP0 HMBADDR MSB	0x91	2	テストのみ
BU WADDR COMP0 HMBADDR MID	0x92	8	
BU WADDR COMP0 HMBADDR LSB	0x93	8	

表C. 3. 2

イメージフォーミング部アドレス発生器キーホール (1/11)

[1901]

[表247]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP1 HMBADDR MSB	0x95	2	テストのみ
BU WADDR COMP1 HMBADDR MID	0x96	8	
BU WADDR COMP1 HMBADDR LSB	0x97	8	
BU WADDR COMP2 HMBADDR MSB	0x99	2	テストのみ
BU WADDR COMP2 HMBADDR MID	0x9a	8	
BU WADDR COMP2 HMBADDR LSB	0x9b	8	
BU WADDR COMP0 VMBADDR MSB	0x9d	2	テストのみ
BU WADDR COMP0 VMBADDR MID	0x9e	8	
BU WADDR COMP0 VMBADDR LSB	0x9f	8	
BU WADDR COMP1 VMBADDR MSB	0xa1	2	テストのみ
BU WADDR COMP1 VMBADDR MID	0xa2	8	
BU WADDR COMP1 VMBADDR LSB	0xa3	8	

表C. 3. 2

イメージフォーミング部アドレス発生器キーホール (2/11)

[1902]

50 [表248]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP2 VMBADDR MSB	0xa5	2	テストのみ
BU WADDR COMP2 VMBADDR MID	0xa6	8	
BU WADDR COMP2 VMBADDR LSB	0xa7	8	
BU WADDR VBADDR MSB	0xa9	2	テストのみ
BU WADDR VBADDR MID	0xaa	8	
BU WADDR VBADDR LSB	0xab	8	
BU WADDR COMP0 HALF WIDTH IN BLOCKS MSB	0xad	2	ロードされ なければ ならない
BU WADDR COMP0 HALF WIDTH IN BLOCKS MID	0xae	8	
BU WADDR COMP0 HALF WIDTH IN BLOCKS LSB	0xaf	8	

表C. 3. 2

イメージフォーマット部アドレス発生器キーホール (3/11)

[1903]

[表249]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP1 HALF WIDTH IN BLOCKS MSB	0xb1	2	ロードされ なければ ならない
BU WADDR COMP1 HALF WIDTH IN BLOCKS MID	0xb2	8	
BU WADDR COMP1 HALF WIDTH IN BLOCKS LSB	0xb3	8	
BU WADDR COMP2 HALF WIDTH IN BLOCKS MSB	0xb5	2	ロードされ なければ ならない
BU WADDR COMP2 HALF WIDTH IN BLOCKS MID	0xb6	8	
BU WADDR COMP2 HALF WIDTH IN BLOCKS LSB	0xb7	8	
BU WADDR HB MSB	0xb9	2	テストのみ
BU WADDR HB MID	0xba	8	
BU WADDR HB LSB	0xbb	8	

表C. 3. 2

イメージフォーマット部アドレス発生器キーホール (4/11)

[1904]

[表250]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP0 OFFSET MSB	0xb d	2	ロードされ なければ ならない
BU WADDR COMP0 OFFSET MID	0xb e	8	
BU WADDR COMP0 OFFSET LSB	0xb f	8	
BU WADDR COMP1 OFFSET MSB	0xc 1	2	ロードされ なければ ならない
BU WADDR COMP1 OFFSET MID	0xc 2	8	
BU WADDR COMP1 OFFSET LSB	0xc 3	8	
BU WADDR COMP2 OFFSET MSB	0xc 5	2	ロードされ なければ ならない
BU WADDR COMP2 OFFSET MID	0xc 6	8	
BU WADDR COMP2 OFFSET LSB	0xc 7	8	
BU WADDR SCRATCH MSB	0xc 9	2	ロードされ なければ ならない
BU WADDR SCRATCH MID	0xc a	8	
BU WADDR SCRATCH LSB	0xc b	8	

表C. 3. 2

イメージフォーマッティング部アドレス発生器キーホール (5/11)

【1905】

【表251】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR MSB WIDE MSB	0xc d	2	ロードされ なければ ならない
BU WADDR MSB WIDE MID	0xc e	8	
BU WADDR MSB WIDE LSB	0xc f	8	
BU WADDR MSB HIGH MSB	0xd 1	2	ロードされ なければ ならない
BU WADDR MSB HIGH MID	0xd 2	8	
BU WADDR MSB HIGH LSB	0xd 3	8	
BU WADDR COMP0 LAST MB IN ROW MSB	0xd 5	2	ロードされ なければ ならない
BU WADDR COMP0 LAST MB IN ROW MID	0xd 6	8	
BU WADDR COMP0 LAST MB IN ROW LSB	0xd 7	8	
BU WADDR COMP1 LAST MB IN ROW MSB	0xd 9	2	ロードされ なければ ならない
BU WADDR COMP1 LAST MB IN ROW MID	0xd a	8	
BU WADDR COMP1 LAST MB IN ROW LSB	0xd b	8	

表C. 3. 2

イメージフォーマッティング部アドレス発生器キーホール (6/11)

【1906】

【表252】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP2 LAST MB IN ROW MSB	0xdd-	2	ロードされ なければ ならない
BU WADDR COMP2 LAST MB IN ROW MID	0xde	8	
BU WADDR COMP2 LAST MB IN ROW LSB	0xdf	8	
BU WADDR COMP0 LAST MB IN HALF ROW MSB	0xe1	2	ロードされ なければ ならない
BU WADDR COMP0 LAST MB IN HALF ROW MID	0xe2	8	
BU WADDR COMP0 LAST MB IN HALF ROW LSB	0xe3	8	
BU WADDR COMP1 LAST MB IN HALF ROW MSB	0xe5	2	ロードされ なければ ならない
BU WADDR COMP1 LAST MB IN HALF ROW MID	0xe6	8	
BU WADDR COMP1 LAST MB IN HALF ROW LSB	0xe7	8	

表C. 3. 2

イメージフォーマッティング部アドレス発生器キーホール (7/11)

【1907】

【表253】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP2 LAST MB IN HALF ROW MSB	0x e 9	2	ロードされ なければ ならない
BU WADDR COMP2 LAST MB IN HALF ROW MID	0x e a	8	
BU WADDR COMP2 LAST MB IN HALF ROW LSE	0x e b	8	
BU WADDR COMP0 LAST ROW IN MB MSB	0x e d	2	ロードされ なければ ならない
BU WADDR COMP0 LAST ROW IN MB MID	0x e e	8	
BU WADDR COMP0 LAST ROW IN MB LSB	0x e f	8	
BU WADDR COMP1 LAST ROW IN MB MSB	0x f 1	2	ロードされ なければ ならない
BU WADDR COMP1 LAST ROW IN MB MID	0x f 2	8	
BU WADDR COMP1 LAST ROW IN MB LSB	0x f 3	8	
BU WADDR COMP2 LAST ROW IN MB MSB	0x f 5	2	ロードされ なければ ならない
BU WADDR COMP2 LAST ROW IN MB MID	0x f 6	8	
BU WADDR COMP2 LAST ROW IN MB LSB	0x f 7	8	

表C. 3. 2

イメージフォーマッティング部アドレス発生器キーホール (8/11)

【1908】

【表254】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP0 BLOCKS PER MB ROW MSB	0xf9	2	ロードされ なければ ならない
BU WADDR COMP0 BLOCKS PER MB ROW MID	0xfa	8	
BU WADDR COMP0 BLOCKS PER MB ROW LSB	0xfb	8	
BU WADDR COMP1 BLOCKS PER MB ROW MSB	0xfd	2	ロードされ なければ ならない
BU WADDR COMP1 BLOCKS PER MB ROW MID	0xfe	8	
BU WADDR COMP1 BLOCKS PER MB ROW LSB	0xff	8	
BU WADDR COMP2 BLOCKS PER MB ROW MSB	0x101	2	ロードされ なければ ならない
BU WADDR COMP2 BLOCKS PER MB ROW MID	0x102	8	
BU WADDR COMP2 BLOCKS PER MB ROW LSB	0x103	8	

表C. 3. 2

イメージフォーマッティング部アドレス発生器キーホール (9/11)

【1909】

【表255】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP0 LAST MB ROW MSB	0x105	2	ロードされ なければならない
BU WADDR COMP0 LAST MB ROW MID	0x106	8	
BU WADDR COMP0 LAST MB ROW LSB	0x107	8	
BU WADDR COMP1 LAST MB ROW MSB	0x109	2	ロードされ なければならない
BU WADDR COMP1 LAST MB ROW MID	0x10a	8	
BU WADDR COMP1 LAST MB ROW LSB	0x10b	8	
BU WADDR COMP2 LAST MB ROW MSB	0x10d	2	ロードされ なければならない
BU WADDR COMP2 LAST MB ROW MID	0x10e	8	
BU WADDR COMP2 LAST MB ROW LSB	0x10f	8	
BU WADDR COMP0 HBS MSB	0x111	2	ロードされ なければならない
BU WADDR COMP0 HBS MID	0x112	8	
BU WADDR COMP0 HBS LSB	0x113	8	

表C. 3. 2

イメージフォーマッティング部アドレス発生器キーホール (10/11)

【1910】

【表256】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP1 HBS MSB	0x115	2	ロードされ なければならない
BU WADDR COMP1 HBS MID	0x116	8	
BU WADDR COMP1 HBS LSB	0x117	8	
BU WADDR COMP2 HBS MSB	0x119	2	ロードされ なければならない
BU WADDR COMP2 HBS MID	0x11a	8	
BU WADDR COMP2 HBS LSB	0x11b	8	
BU WADDR COMP0 MAXHB	0x11f	2	ロードされ なければならない
BU WADDR COMP1 MAXHB	0x123	2	
BU WADDR COMP2 MAXHB	0x127	2	
BU WADDR COMP0 MAXVB	0x12b	2	ロードされ なければならない
BU WADDR COMP1 MAXVB	0x12f	2	
BU WADDR COMP2 MAXVB	0x133	2	

表C. 3. 2

イメージフォーマッティング部アドレス発生器キーホール (11/11)

キーホールレジスタは広く2つの範囲に分かれる。それ 50 は、アドレス計算の前にピクチャサイズパラメータをロ

ードしなければならないものと、様々な（水平及び垂直の）ブロック及びマクロブロックカウントの現在の合計を含むものである。ピクチャサイズパラメータはライトアドレス発生器により発生される割り込みに答えてロードされ得る、つまり、ピクチャサイズまたはサンプリングトークンのいずれかがデータストリームに現れる時にロードされ得る。あるいは、ピクチャサイズがデータストリームを再生する前に知られていない場合、リセット後にそれらを書き込むことができる。セッティングの例はセクション C. 13 に記すが、ピクチャサイズパラメータレジスタについては次のセクションにおいて定義する。

【1911】 C. 3. 4 「ライトアドレス発生器のプログラミング」

次のデータバスレジスタは、アドレス計算が進行される前に、正しいピクチャサイズ情報を含んでいなければならない。それらを図 171 に図示する。

【1912】 1) WADDR HALF WIDTH IN BLOCKS: これはブロックにおいて、入ってくるピクチャの半分の幅を定義する。

【1913】 2) WADDR MBS WIDE: これはマクロブロックにおいて、入ってくるピクチャの幅を定義する。

【1914】 3) WADDR MBS HIGH: これはマクロブロックにおいて、入ってくるピクチャの高さを定義する。

【1915】 4) WADDR LAST MB IN ROW: これはマクロブロックの 1 つの全幅のローにおける最後のマクロブロックの左上位のブロックのブロック数を定義する。ブロックナンバリングは左端のマクロブロックの左上コーナーの 0 からスタートし、フレームを横切って各ブロック毎に増加し、マクロブロックロー内の次のブロック・ローへと続く。

【1916】 5) WADDR LAST MB IN HALF ROW: これは前のアイテムと同様であるが、マクロブロックの半幅のローにおける最後のマクロブロックの左上位のブロックのブロック数を定義する。

【1917】 6) WADDR LAST ROW IN MB: これはマクロブロックのロー内の最後のブロックのローにおける左端のブロックのブロック数を定義する。

7) WADDR BLOCKS PER MB ROW: これはマクロブロックの 1 つの全幅のローに含まれる全ブロック数を定義する。

【1918】 8) WADDR LAST MB ROW: これはピクチャ内のマクロブロックの最後のローにおける左端のマクロブロックの左上位のブロックアドレスを定義する。

【1919】 9) WADDR HBS: これは入ってくるピクチャのブロックの幅を定義する。

【1920】 10) WADDR MAXHB: これは 1 つのマクロブロック内のブロックのローにおける右端のブロックのブロック数を定義する。

【1921】 11) WADDR MAXVB: これはブロックにおける、1 つのマクロブロックの高さ - 1 を定義する。

【1922】 それに加えて、DRAM の組織を定義するレジスタをプログラムしなければならない。このレジスタは 3 バイトのレジスタであり、この場合、n はデータストリームにおいて予測される成分数である（それはデータストリームにおいて定義することができ、1 最小限及び 3 最大限であり得る）。

【1923】 多くのパラメータがブロック数もしくはブロックアドレスを指定することに注意。これは最終アドレスがブロックアドレスであると予測され、計算は累積アルゴリズムに基づくからである。

【1924】 図 171 に図示するスクリーン構成は次のレジスタバリューを生じる：

1) WADDR HALF WIDTH IN BLOCKS = 0 x 16

2) WADDR MBS WIDE = 0 x 16

3) WADDR MBS HIGH = 0 x 12

4) WADDR LAST MB IN ROW = 0 x 2A

5) WADDR LAST MB IN HALF ROW = 0 x 14

6) WADDR LAST ROW IN MB = 0 x 2C

7) WADDR BLOCKS PER MB ROW = 0 x 58

8) WADDR LAST MB ROW = 0 x 5D8

9) WADDR HBS = 0 x 2C

10) WADDR MAXVB = 1

11) WADDR MAXHB = 1

C. 3. 5 ステートマシンのオペレーション

バッファマネージャのステートマシンには 19 のステートがあり、詳細は表 257、表 258 に記す。これらは図 173 に示すように、また行動解説、bml og i c. M において説明するように相互作用する。

【1925】

【表 257】

ステート	バリュー
IDLE	0x00
DATA	0x10
CODING STANDARD	0x0C
HORZ MBS0	0x07
HORZ MBS1	0x06
VERT MBS0	0x0B
VERT MBS1	0x0A
OUTPUT TAIL	0x08
HB	0x11
MB0	0x10
MB1	0x12
MB2	0x1E
MB3	0x13
MB4	0x0E
MB5	0x14
MB6	0x15

表C. 3. 3 ライトアドレス発生器のステート (1/2)

[1926]

[表258]

ステート	バリュー
MB4A	0x18
MB4B	0xC9
MB4C	0x17
MB4D	0x16
ADDR1	0x19
ADDR2	0x1A
ADDR3	0x1B
ADDR4	0x1C
ADDR5	0x03
HSAMP	0x05
VSAMP	0x04
PIC ST1	0x0f
PIC ST2	0x01
PIC ST3	0x02

表C. 3. 3 ライトアドレス発生器のステート (2/2)

```

BU WADDR SCRATCH=BU BUFFERn BASE
                    +BU COMPm OFFSET;
BU WADDR SCRATCH=BU WADDR SCRATCH
                    +BU WADDR VMBADDR;
BU WADDR SCRATCH=BU WADDR-SCRATCH
                    +BU WADDR HMBADDR;
BU WADDR SCRATCH=BU+WADDR SCRATCH
                    +BU WADDR VBADDR;
out addr=BU WADDR SCRATCH
                    +BU WADDR HB;

```

1) BU WADDR VMBADDR: マクロブロックのローの左端のマクロブロックの(左上位のブロック)ブロックアドレス、そこにアドレスが計算されるブロックが含まれる。

[1927] 2) BU WADDR HMBADDR: マクロブロックのカラムの上位マクロブロックの(左上位のブロック)ブロックアドレス、そこにアドレスが計算されるブロックが含まれる。

C. 3. 5. 1 「アドレスの計算」

ライトアドレス発生器ステートマシンの主セクションについては、図173の左側に図示している。データトークンを受け取ると、ステートマシンはステートIDLEからステートADDR1に、そして順にステートADDR5へと移動し、そこから2線式インターフェースの制御で18ビットのブロックアドレスを出力する。ステートADDR1~ADDR5により遂行される計算は以下の通りである:

[1928] 3) BU WADDR VBADDR: ブロックのローの左端の、マクロブロックロー内のブロックアドレス、そこにアドレスが計算されるブロックが含まれる。

[1929] 4) BU WADDR HB: アドレスが計算されるブロックの、マクロブロック内の水平ブロック数。

[1930] 5) BU WADDR SCRATCH

585

中間結果の一時的記憶のために使用されるスクラッチレジスタ。

【1931】図172を考慮し、例えば、そのアドレスが $0 \times 62D$ であるブロックの計算を取り上げると、以下の計算シーケンスが発生する：

スクラッチ=BUFFERn BASE+COMPm
OFFSET; (0と仮定)

スクラッチ= $0 + 0 \times 5D8$;

スクラッチ= $0 \times 5D8 + 0 \times 28$;

スクラッチ= $0 \times 600 + 0 \times 2C$;

ブロックアドレス= $0 \times 62C + 1 = 0 \times 62D$;

様々なレジスタの内容を図に示す。

【1932】C. 3. 5. 2 「新しいスクリーンロケーションパラメータの計算」

アドレスが出力されたら、ステートマシンは上記の様々なスクリーンロケーションパラメータを更新するために計算を実行し続ける。ステートHB及びMB0~MB6が計算を行い、あるポイントにおいてステートデータに制御を伝送し、ステートデータからデータトークンの残りが出力される。

【1933】これらのステートはペアで進行し、第1のペアは現在のカウンとその終末値との差を計算し、0

ステートHB及びMB0:

スクラッチ= $hb - \max hb$;

if (z)

hb=0;

else

{

hb=hb + 1

new state=DATA;

}

ステートMB1及びMB2:

スクラッチ=vb addr-last row in mb;

if (z)

vb addr=0;

else

{

vb addr=vb addr + width in blocks;

new state=DATA;

}

ステートMB3及びMB4:

スクラッチ=hmb addr-last mb in row;

if (z)

hmb addr=0;

else

{

hmb addr=hmb addr + maxhb;

new state=DATA;

ステートMB5及びMB6:

586

フラグを発生させる。第2のペアはレジスタをリセットするか、あるいは(スクリーンサイズから引き出されるセットアップレジスタにおけるバリュエに基づいて)固定オフセットを加算する。いずれの場合にも、考慮中のカウンとその終末値に達すると(つまり、0フラグが設定されると)、制御はステートのMBシーケンスを下方へと続ける。そうでなければ、全てのカウンは正しい(次のアドレス計算の準備が整っている)とみなされ、制御はステートデータを伝送する。

10 【1934】加算もしくは減算の使用を含む全てのステートは、完了するため(標準のリップル・キャリアダーの使用を許す)2つのサイクルを取り、これはアダプターのステートのために1と0の間で交替するフラグ、fc(最初のサイクル)の使用により達成される。

【1935】アドレス計算及びスクリーンロケーション計算ステートの全てが、好ましい2線式インターフェース条件を仮定して、データが出力されるようにする。

【1936】C. 3. 5. 2. 1 「規格用の計算(MPEG-スタイル)シーケンス」

20 オペレーションのシーケンスは以下の通りである(0フラグはアダプターの出力に基づく):

587

588

```
スクラッチ=vmb addr-last mb row;
```

```
if (!z)
```

```
    vmb addr=vmb addr
```

```
        + blocks per mb row;
```

(vmb addrは、ピクチャエンドが計算から推論される時より、ピクチャスタートトークンが検出された後に、リセットされる。)

C. 3. 5. 2. 2 「H. 261用計算シーケンス」
H. 261計算用シーケンスはステートMB4における標準シーケンスから分岐する：

ステートHB及びMB0：上記の通り

ステートHB1及びMB2：上記の通り

ステートHB3及びMB4：

```
スクラッチ=hmb addr-last mb in row;
```

```
if (z & (mod 3 == 2)) /*end of slice on  
                        right of screen*/
```

```
{
```

```
    hmb addr=0;
```

```
    new state=MB5;
```

```
}
```

```
else if (z) /*end of row on right  
            of screen*/
```

```
{
```

```
    hmb addr=half width in blocks;
```

```
    new state=MB4A;
```

```
}
```

```
else
```

```
{
```

```
スクラッチ=hmb addr
```

```
    -last mb in half row;
```

```
new state=MB4B;
```

```
}
```

ステートMB4A：

```
vmb addr=vmb addr
```

```
    + blocks per mb row;
```

```
new state=DATA;
```

ステート (MB4) 及びMB4B：

```
(scratch=hmb addr
```

```
    -last mb in half row;)
```

```
if (z & (mod 3 == 2)) /*end of slice on  
                    left of screen*/
```

```
{
```

```
    hmb addr=hmb addr + maxhb;
```

```
    new state=MB4C;
```

```
}
```

```
else if (z) /*end of row on left  
            of screen*/
```

```
{
```

```
    hmb addr=0;
```

```
    new state=MB4A;
```

```
}
```

```
else
```

```
{
```

589

590

```
hmb addr=hmb addr + maxhb;
new state=DATA;
```

}

ステートMB4C及びMB4D:

```
vmb addr=vmb addr
      -blocks per mb row;
vmb addr=vmb addr
      -blocks per mb row;
new state=DATA;
```

ステートMB5及びMB6:上記の通り

C. 3. 5. 3 「ピクチャスタートトークン上のオペレーション」

ピクチャスタートトークンを受け取ると、制御はステートPIC ST1にに進み、そこでvb addrレジスタ (BU WADDR VBADDR) が0にリセットされる。ステートPIC ST2及びPIC ST3の各々が各成分のために一度づつ巡視され、hmb addrとvmb addrを各々リセットする。次に制御はステートOUTPUT TAILを介してIDLE

20

C. 3. 5. 4 「DEFINE SAMPLINGトークン上のオペレーション」

DEFINE SAMPLINGトークンを受け取ると、成分レジスタには最も重要でない2ビットの入力データがロードされる。それに加えて、ステートHSAMP及びVSAMPを介して、その成分用のmaxhbレジスタ及びmaxvbレジスタがロードされる。更に、適切な定義サンプリングイベントビットがトリガーされる (全トークンが書き込まれるようにするため1サイクル

30

【1937】C. 3. 5. 5 「HORIZONTAL MBS及びVERTICAL MBS上のオペレーション」

HORIZONTAL MBS及びVERTICAL MBSの各々が到着すると、トークンに含まれる14ビットバリューが2つのサイクルで適切なレジスタに書き込まれる。関連イベントビットがトリガーされ、1サイクル分遅延される。

【1938】C. 3. 5. 6 「他のトークン」

コーディングスタンダードトークンが検出され、トップレベルのBU WADDR COD STDレジスタに入力データが書き込まれるようにする。これがデコードされ、(H261ではなく) nh261フラグがバッファマネージャブロックにハードワイヤードされる。他の

40

全てのトークンが制御をステートOUTPUT TAILに動かし、トークンが完了するまでそのステートがデータをアクセプトする。しかしながら、それは如何なるデータも実際には出力しないことに注意。

【1939】セクションC. 4 「リードアドレス発生器」

C. 4. 1 「展望」

本発明のリードアドレス発生器は4つのステートマシン/データバスブロックから成る。まず、dlineがラインアドレスを発生させ、それらを他の3つの (各成分用に1つづつの) 同じページ/ブロックアドレス発生器、dramctlsに分配する。全てのブロックは2線式インターフェースによってつながれる。オペレーションモードは飛び越し/順送り型、最初のフィールド上位/下位、及び上位/下位/両方のフレームスタートの全ての組合せを含む。表259、表260はdispaddr制御レジスタの名前、アドレス、及びリセットステートを示し、セクションC. 13において両アドレス発生器用のプログラミングの例を示す。

【1940】C. 4. 2 「ラインアドレス発生器 (dline)」

このブロックは各成分のためにラインアドレス発生器を計算する。表259、表260はdlineにおける18ビットのデータバスレジスタを示す。

【1941】DISP register name及びADDR register name DISP nameレジスタの区別はdispaddrのみにあり、DRAMから読み出される表示エリアに対してそのレジスタが特別であることを意味する。ADDR nameはそのレジスタが外部バッファの構造に関する何かを説明することを意味する。

【1942】オペレーション

リピート等の全てのモードを無視する、dlineの基本的オペレーションは:

```
if (vsync start) /*first active
cycle of vsync*/
{
  comp=0;
  DISP VB CNT COMP[comp]=0;
```

591

592

```
LINE [comp] = BUFFER BASE [comp] + 0 ;
LINE [comp] = LINE [comp]
                + DISP COMP OFFSET [comp] ;
while (VB CNT COMP [comp]
        < DISP VBS COMP [comp])
{
while (line count [comp] < 8)
{
{
while (comp < 3)
{
→ OUTPUT LINE [comp] to dramctl [comp]
line [comp] = LINE [comp]
                + ADDR HBS COMP [comp] ;
comp = comp + 1 ;
}
line count [comp] = line count [comp]
                    + 1 ;
}
VB CNT COMP [comp] = VB CNT COMP [comp]
                    + 1 ;
line count [comp] == 0 ;
}
}
```

[1943]

【表259】

レジスタ名	バス	キーホール アドレス	説明	コメント
BUFFER BASE0	A	0x00, 01, 02, 03	各バッファ スタートの アドレス	これらのレジスタは オペレーションを始 める前にリセットさ ねばならない。
BUFFER BASE1	A	0x04, 05, 06, 07		
BUFFER BASE2	A	0x08, 09, 0a, 0b		
DISP COMP OFFSET0	B	0x24, 25, 26, 27	バッファ ベースから リーディング 場所までの オフセット	
DISP COMP OFFSET1	B	0x28, 29, 2a, 2b		
DISP COMP OFFSET2	B	0x2c, 2d, 2e, 2f		
DISP VBS COMP0	B	0x30, 31, 32, 33	読まれるべ き垂直プロ ット数	
DISP VBS COMP1	B	0x34, 35, 36, 37		
DISP VBS COMP2	B	0x38, 39, 3a, 3b		

表C. 3. 4 Dispaddrデータバスレジスタ (1/2)

[1944]

[表260]

レジスタ名	バス	キーホール アドレス	説明	コメント
ADDR HBS COMP0	B	0x3c, 3d, 3e, 3f	DATA内 の水平プロ ック数	これらのレジスタは オペレーションを始 める前にリセットさ れなければならない。
ADDR HBS COMP1	B	0x40, 41, 42, 43		
ADDR HBS COMP2	B	0x44, 45, 46, 47		
LINE0	A	0x0c, 0d, 0e, 0f	現在のライ ンアドレス	これらのレジスタは dispaddrに より使用される一時的なレジスタである。 注: 全てのレジスタはR/Wである。
LINE1	A	0x10, 11, 12, 13		
LINE2	A	0x14, 15, 16, 17		
DISP VB CNT COMP0	A	0x18, 19, 1a, 1b	今も読まれ るべき垂直 ブロック数	
DISP VB CNT COMP1	A	0x1c, 1d, 1e, 1f		
DISP VB CNT COMP2	A	0x20, 21, 22, 23		

表C. 3. 4 Dispaddrデータバスレジスタ (2/2)

C. 4. 3 「Dline制御レジスタ」

[1945]

上記オペレーションは下記の表に示すdispaddr

[表261]

制御レジスタにより修正される。

レジスタ名	アドレス	ビット	リセット ステート	機能
LIES IN LAST ROW0	0x08	[2:0]	0x07	これらの3つの レジスタは読み出 されるブロックの 最後のローの(8 の内の)ライン数 を決定する。
LIES IN LAST ROW1	0x09	[2:0]	0x07	
LIES IN LAST ROW2	0x0a	[2:0]	0x07	
DISPADDR ACCESS	0x0b	[0]	0x00	dispaddr のアクセス ビット
DISPADDR CTL0 これらの制御ビット の詳細な説明に ついては下記を参照 せよ。	0x0c	[1:0]	0x00	SYNC MODE
		[2]	0x00	READ START
		[3]	0x1	INTER- LACED/ PROG
		[4]	0x0	LSB INVERT
		[7:5]	0x0	LINE PAT
DISPADDR CTL1	0x0d	[0]	0x1	COMP0- HOLD

C. 4. 3. 1 「LINES IN LAST ROW [成分]」

これら3つのレジスタは、各成分のために、読まれることになっているブロックの最後のローの中のライン数を決定する。このように、リードウィンドーの高さは任意のライン数であってよい。ウィンドーの上端、左端及び右端がブロック境界上にあり、出力コントローラが余分なラインをクリップする(捨てる)ことができるので、
30 これはバックアップの特徴である。

【1946】C. 4. 3. 2 「DISPADDR ACCESS」

これはdispaddr全体のためのアクセスビットである。このロケーションに「1」を書き込むと、dispaddrがクロックに同期して停止される。このステートに達すると、全てのdispaddrレジスタに対する非同期upiアクセスを遂行しても安全である。アクセスビットが「1」になるまで、upiがデータバスレジスタから能動的に締め出されることに注意。現在の表示またはデータバスオペレーションを崩壊させることなくdispaddrに対するアクセスを達成するために、アクセスは以下の状況下でのみ与えられ、リリースされる。

【1947】ストップping: データバスがその現在の2サイクルオペレーションを完了した場合(1回のオペレーションを行った場合)、そして出力コントローラからの「安全」信号が高い場合にのみアクセスが認められるであろう。この信号は表示ウィンドーの下スクリーン上のエリアを表し、出力コントローラ(dispaddr) 50

ではない) においてプログラムされる。従って、dispaddrに対するアクセスを得ようとする前に、出力コントローラをプログラムすることが必要であることに注意。

【1948】スターティング・アクセスは「安全性」が高い時、もしくはvsyncの間にのみリリースされるであろう。これは表示が活性ウィンドーにあまりに近付いてスタートしないことを保証する。

【1949】このスキームは制御ソフトウェアがアクセス、ディスプレイ・エンドまで登録、dispaddrを修正、及びアクセスをリリース等をリクエストできるようにする。ソフトウェアがあまりに遅く、vsyncの後までアクセスビットをリリースしない場合、dispaddrは次の安全な期間までスタートしないであろう。ボーダーカラーは(がらくたというよりむしろ)この「失われた」ピクチャの間に表示されるであろう。

C. 4. 3. 3 「DISPADDR CTL0 [7:0]」

以下の説明を読む際に、飛び越しデータと飛び越し表示の間の区別を理解することが重要である。

【1950】飛び越しデータには2つの形態があり得る。トップレベルレジスタはフィールド・ピクチャ(各々のパッファが1つのフィールドを含む)とフレーム(飛び越されていようとなかろうと、各々のパッファが全体のフレームを含む)を支持する。

【1951】DISPADDR CTL0 [7:0] は以下の制御ビットを含む:
SYNC MODE [1:0]

599

飛び越し表示では、トップからボトムフィールドに言及する vsyncs は field info ピンによって区別される。このような状況では、field info = HIGH がトップフィールドを意味する。これら 2 つの制御ビットが、どの vsyncs dispaddr がバッファマネージャから新しいディスプレイバッファをリクエストするかを決定し、こうして（データが飛び越された場合）バッファ内のフィールドをディスプレイ上のフィールドと同期化する：

- 0 : トップフィールド上の新しいディスプレイバッファ
- 1 : ボトムフィールド
- 2 : 両フィールド
- 3 : 両フィールド

スタートアップ時に、dispaddr はあらゆる vsync 上のバッファマネージャからバッファを要請するであろう。バッファが準備できるまで、dispaddr は 0（ノーディスプレイ）バッファを受け取るであろう。最終的に良いバッファインデックスを受け取ると、dispaddr はそれがディスプレイ上のどこにあるかに関して何の知識も持っていない。従って、ディスプレイスタートアップを正しい vsync と同期させる必要があるかもしれない。

【1952】READ START

スタートアップ時の飛び越し表示のために、このビットはどの vsync ディスプレイ上で実際スタートするかを決定する。更に、ディスプレイバッファインデックスを受け取った後、dispaddr はディスプレイ上のフィールドをバッファ内のフィールドと整列させるために、現在の vsync を「居残らせ」てもよい。

【1953】INTERLACED/PROGRESSIVE

- 0 : Progressive (順送り型)
- 1 : Interlaced (飛び越し)

順送りモードでは、全てのラインがバッファのディスプレイエリアから読まれる。飛び越しモードでは、交互のラインが読まれる。リーディングが最初のラインからスタートするか、二番目のラインからスタートするかは field info 次第である。（飛び越し）フィールド・ピクチャでは、システムが各バッファからの全てのラインを読みたいと望むので、このビットのセッティングは順送り型であろうということに注意。field info と第 1/第 2 のライン間のマッピングは lsb invert（歴史的な理由からそう名付けられた）により反転されてもよい。

【1954】LSB INVERT

```
while (true)
```

```
{
  CNT_LEFT=0;
  GET A NEW LINE ADDRESS from dline;
  BLOCK_ADDR=input_block_addr+0;
```

600

設定されると、このビットはラインカウンタによって見られる field info 信号を反転する。このように、リーディングはフレームの正しいライン上でスタートされ、エンコーダ、ディスプレイ、またはトップレベルレジスタが採用する規定に関わりなく、ディスプレイに整列させられてもよい。

【1955】LINE RPT [2:0]

設定されると、各々のビットは対応する成分のラインを二度読ませる（ビット 0 が成分 0 に影響を与える等）。これは垂直のアンサンプリングの最初の部分を形成する。それは QFIF から 601 への変換のために要求される 8 回の色アンサンプリングに使用される。

【1956】COMPOHOLD

このビットは成分 1 及び 2 のライン数に対する、成分 0 のため（表示されるものに対立するものとして）読まれるライン数の比率をプログラムするために使用される。

【1957】0 : 同じライン数、つまりバッファ内の 4 : 4 : 4 のデータ

1 : 成分 0 ラインの二倍、つまり 4 : 2 : 0。

【1958】ページ/ブロックアドレス発生器 (dramctl s)

ラインアドレスを通過する時、これらのブロックは一連のページ/ラインアドレス、及びラインに沿って読むべきブロックを発生させる。8 ブロックの最低ページ幅が常に仮定され、結果的に生じる出力はページアドレス、3 ビットライン数、3 ビットブロックスタート、及び 3 ビットブロックストップアドレスで構成される。（ライン数は dline によって計算され、無修正の dramctl s を通して送られる。）このように、ライン 5 の 48 ピクセルを読み取ることは、左から 3 番目のブロック（任意のラインに沿った任意のポイント）から始まるページ 0 xaa を形成し、DRAM インターフェースに送られるアドレスは以下の通りであろう：

ページ = 0 xaa

ライン = 5

ブロック・スタート = 2

ブロック・ストップ = 7

これら 3 つのマシンの各々は 5 個のデータバスレジスタを持つ。これらを表 259、表 260 に示す。各 dramctl の基本行動は以下の通りである：

ブロック・スタート = 2

ブロック・ストップ = 7

これら 3 つのマシンの各々は 5 個のデータバスレジスタを持つ。これらを表 259、表 260 に示す。各 dramctl の基本行動は以下の通りである：

601

602

```

PAGE ADDR=input page addr+0;
CNT LEFT=DISP HBS+0;
while (CNT LEFT>BLOCKS LEFT)
{
BLOCKS LEFT=8-BLOCK ADDR;
→output PAGE ADDR, start=BLOCK ADDR
stop=7.
PAGE ADDR=PAGE ADDR+1;
BLOCK ADDR=0;
CNT LEFT=CNT LEFT-BLOCKS LEFT;
}
/*Last Page of line*/
CNT LEFT=CNT LEFT+BLOCK ADDR;
CNT LEFT=CNT LEFT-1;
→ output PAGE ADDR,
start=BLOCK ADDR, stop=CNT LEFT
}

```

【1959】

【表262】

レジスタ名	バス	キーホール アドレス	説明	コメント
DISP COMP0 HBS	A	0x48, 49, 4a, 4b	読まれるべき 水平プロ ック数	このレジスタはオペ レーションを開始す る前にロードされな ければならない。
DISP COMP1 HBS	A	0x4c, 4d, 4e, 4f	ADDR HBS参照	
DISP COMP2 HBS	A	0x50, 51, 52, 53		
CNT LEFT0	A	0x54, 55, 56, 57	今も読まれ るべきプロ ック数	これらのレジスタは disp.addrに よって使用される一 時的なレジスタであ る。 注：全てのレジスタ は初期化される。
CNT LEFT1	A	0x58, 59, 5a, 5b		
CNT LEFT2	A	0x5c, 5d, 5e, 5f		
PAGE ADDR0	A	0x60, 61, 62, 63	現在の頁の アドレス	
PAGE ADDR1	A	0x64, 65, 66, 67		
PAGE ADDR2	A	0x68, 69, 6a, 6b		

表C. 3. 5

Dramctl (0, 1&2) データバスレジスタ (1/2)

【1960】

【表263】

レジスタ名	バス	キーホール アドレス	説明	コメント
BLOCK ADDR0	B	0x6c, 6d, 6e, 6f	現在のブロッ クアドレ ス	これらのレジスタは dispaddrに より使用される一瞬 的なレジスタ である。 注：全てのレジスタ はCバスからロー ドされる。
BLOCK ADDR1	B	0x70, 71, 72, 73		
BLOCK ADDR2	B	0x74, 75, 76, 77		
BLOCKS LEFT0	B	0x78, 79, 7a, 7b	現在の頁に 残された ブロック	
BLOCKS LEFT1	B	0x7c, 7d, 7e, 7f		
BLOCKS LEFT2	B	0x80, 81, 82, 83		

表C. 3. 5

Dramctl (0, 1&2) データバスレジスタ (2/2)

プログラミング

次の15のdispaddrレジスタはオペレーションを開始する前にプログラムされなければならない。

【1961】

BUFFER BASE0, 1, 2
DISP COMP OFFSET0, 1, 2
DISP VBS COMP0, 1, 2
ADDR HBS COMP0, 1, 2
DISP COMP0, 1, 2 HBS
dispaddr制御レジスタのリセットステートの使用は、4:2nの飛び越しディスプレイを与え、ライン
リピートは同期化されず、トップフィールド (field
info=HIGH) でスタートするであろう。図
168、「SIF (22x18のマクロブロック) ピク
チャを含むバッファ0」は典型的なSIFピクチャ用の
バッファセットアップを示す。(この例については、
C. 13章で詳細に説明する)。この例において、DI
SP HBS COMPnはADDR HBS COM
Pnに等しく、同様に垂直のレジスタDISP VBS
COMPn及び等価ライトアドレス発生器レジスタが等
しい、つまり、読まれるべきエリアは全バッファである
ことに注意。

【1962】リードアドレス発生器を用いてのウインド
ー処理

バッファの一部 (ウインドー) のみを読むようにdis
paddrをプログラムすることが可能である。ウイン
ドーのサイズは、レジスタDISP HBS、DISP
VBS、COMPONENT OFFSET、及びL
INES INLAST ROWにより各成分のために
プログラムされる。図169、「ディスプレイウインド
ーを備えたSIF成分0」は、これが如何にして (成分

20 0だけのために) 達成されるかを示している。

【1963】この例では、レジスタセッティングは以下
の通りであろう：

BUFFER BASE0=0x00
DISP COMP OFFSET0=0x2D
DISP VBS COMP0=0x22
ADDR HBS COMP0=0x2C
DISP HBS COMP0=0x2A

注：

・ウインドーはブロック境界上でのみスタート及びスト
ップできる。本例では、7に等しいLINES IN
LAST ROWを残した (全てが8であることを意味
する)。

【1964】・本例は4:4:4のデータ以外のものでは
は実用的ではない。調和するために、他の2つの成分の
ためのウインドーエッジはブロック境界上に存在するこ
とができないであろう。

【1965】・色空間変換はそれが受け取るデータが
4:4:4ではない場合に停止するであろう。これはこ
れらのリードウインドーがアンサンブラと共に、これを
達成するようにプログラムされなければならないことを
意味する。

セクションC. 5 「アドレス発生用のデータバス」
dispaddr及びwaddrgenにおいて使用さ
れるデータバスは構造及び幅 (18ビット) が同じであ
り、レジスタ数、ある種のマスキング、及びステートマ
シーンに戻されるフラグが異なるだけである。1スライ
スの回路を図174、「データバスのスライス」に示
す。レジスタは唯ードライブAまたはBバスに指定さ
れ、それらはコントローラにおいて最大限活用されて、
使用 (指定) される。全てのレジスタはCバスからロー

ド可能であるが、全ての「ロード」信号が駆動されるわけではない。アダgerを含む全てのオペレーションはアダgerに通常の波及的桁上げを持たせる2つのサイクルをカバーする。図175、「データバスの2サイクルオペレーション」において、Aバスレジスタにロードバックされる2個のレジスタの2個のサイクルの合計に対するタイミングを示す。様々なフラグがc c o d e発生を可能にするために、データバス内でp t o化され、同じ理由から、データバス減略図の構成は少しばかり異なる。(A及びBバス上への)全てのレジスタ用のトライ
10 スデータは、セル内の結合バスを除去する1つのブロックにおいてであるので、より良いc c o d e発生を可能にする。データバスに対するu p iアクセスを得るために、アクセスビットはこれがなければu p iがロックアウトされるように設定されなければならない。U p iアクセスはリード及びライトとは異なる：

・ライティング：アクセスビットが設定されると、全てのロード信号は不能化され、一連の3バイトアドレスドライブストロブの1つがレジスタの1つの適切なバイトに動かされる。U p iデータバスはデータバスを垂直
20 に進み(複写される、2-8-8ビット)、18ビットレジスタが3個の別のバイトライトとして書き込まれる。

【1966】・リーディング：これはA及びBバスを使用して行われる。ここでも、アクセスビットが設定されなければならない。アドレスレジスタはAまたはBバス上に動かされ、u p iバイトセレクトが関連バスからバイトを取り上げ、それをu p iバス上へと動かす。

【1967】ダブルサイクルデータバスオペレーションがA及びBバスにそれらの値を保持するように要求し、
30 u p iアクセスがこれらを崩壊させるので、アクセスはいずれかのデータバスオペレーションのスタート前に、ステートマシンを制御することによってのみ与えられなければならない。

【1968】両アドレス発生器内の全てのデータバスレジスタは、トップレベルアドレス0 x 2 8 (m s b)の9ビット幅のキーホールを通して、キーホール用には0 x 2 9 (l s b)であり、データ用には0 x 2 Aを通してアドレスされる。キーホールアドレスを表282~表300に示す。

【1969】注：

1) アドレス発生器(d i s p a d d rとw a d d r g e n)内の全てのアドレスレジスタはブロック化アドレスを包含する。ピクセルアドレスは決して使用されず、ラインアドレスを包含する唯一のレジスタは、3個のL I N E S I N L A S T R O Wレジスタである。

【1970】2) 幾つかのレジスタはアドレス発生器間で重複される、例えば、B U F F E R B A S E 0はd i s p a d d rとw a d d r g e n用のアドレススペースにおいて発生する。これらは両方共ローディングを必
50

要とする2個の別々のレジスタである。これにより、表示ウインドー処理(表示記憶装置の一部だけを読む)が可能となり、3成分ビデオ以外のフォーマット表示が容易になる。

【1971】セクションC. 6 「DRAMインターフェース」

C. 6. 1 「展望」

本発明においては、空間デコーダ、時間デコーダ、及びビデオフォーマッティング部がその特別なチップのために各々データトークンブロックを含む。3個全ての装置において、D R A Mインターフェースの機能はチップから外部D R A Mへ、そして外部D R A Mからチップへと、アドレス発生器により供給されるブロックアドレスを介してデータを伝送することである。

【1972】D R A Mインターフェースは典型的に、アドレス発生器に対して、そしてそれを通してデータが送られる様々なブロックのクロックに対して、非同期であるクロックから操作する。しかしながら、この非同期性はクロックがほぼ同じ周波数で操作するので、容易に
処理される。

【1973】データは通常D R A Mインターフェースと64バイトのブロック内の残りのチップとの間で伝送される(唯一の例外は時間デコーダ内の予測データである)。伝送は「スイングバッファ」として知られる装置によって行われる。これは本質的にダブルバッファされた配置で操作される一対のR A Mであり、D R A Mインターフェースが1つのR A Mを詰めるか空にしている間に、チップの他の部分が他のR A Mを空にするか詰めている。アドレス発生器からアドレスを運ぶ別のバスが各々のスイングバッファと連合する。

【1974】各々のチップは4つのスイングバッファを持つが、これらのスイングバッファの機能は各々の場合により異なる。空間デコーダにおいて、コード化データをD R A Mに伝送するために1つのスイングバッファが使用され、D R A Mからコード化データを読むために別のスイングバッファが使用され、トークン化データをD R A Mに伝送するために3番目のスイングバッファが、またD R A Mからトークン化データを読むために4番目のスイングバッファが使用される。時間デコーダにおいては、イントラもしくは予測されたピクチャデータをD R A Mに書き込むために1つのスイングバッファが使用され、D R A Mからイントラもしくは予測されたピクチャデータを読むために2番目のスイングバッファが使用され、他の2つはフォワード及びバックワード予測データを読むために使用される。ビデオフォーマッティング部においては、1つのスイングバッファがD R A Mにデータを伝送するために使用され、他の3つがD R A Mからデータを読むために使用されるが、それは各々輝度(Y)、及び赤と青の色差データ(各々C r及びC b)
である。

【1975】DRAMインターフェースの一般的な特徴のオペレーションについては、空間デコーダ文書において記載している。次のセクションは本発明によるDRAMインターフェースの特徴、特にビデオフォーマッティング部に特有の特徴について説明する。

【1976】C. 6. 2 「ビデオフォーマッティング部DRAMインターフェース」

ビデオフォーマッティング部において、データはブロックで外部DRAMに書き込まれるが、ランダムオーダーで読み出される。ライティングは空間デコーダに関して既に説明したものと正確に同じであるが、リーディングは少々複雑である。

【1977】ビデオフォーマッティング部外部RAM内のデータは少なくとも8ブロックのデータが1ページに収まるように組織化される。これら8ブロックは8個の連続的な水平ブロックである。ラスタ化する際に、8個の連続ブロックの各々から8バイトを読み出し、それをシングバッファに書き込む（つまり、8ブロックの各々に同じロー）ことが必要である。

【1978】トップフローを考えてみる（そしてバイト幅のインターフェースを仮定する）と、xアドレス（3LSB）が0に設定され、yアドレス（3MSB）も同様である。xアドレスは次に、最初の8バイトの各々が読み出されるにつれて増分される。この時点で、アドレスの上位部分（ビット6以上—LSB=ビット0）が増分され、xアドレス（3LSB）は0にリセットされる。このプロセスは64バイトが読み出されるまで繰り返される。外部DRAMに対する16または32ビット幅のインターフェースでは、xアドレスは1の代わりに2または4だけ単に増分される。

【1979】アドレス発生器は8バイトの倍数が常に読まれるが、64以下のバイトが読まれるべきである（これはラスタラインの始まりまたは終わりにおいて要求される）という信号をDRAMインターフェースに送ることができる。これはスタート及びストップバリューを使用して行われる。スタートバリューはアドレスの上位部分（6ビット以上）のために使用され、ストップバリューがこれと比較され、リーディングをストップすべきであることを指示する信号が発せられる。

【1980】セクションC. 7 「垂直アップサンプリング」

C. 7. 1 「序文」

1つの色成分のピクセルのラスタースキャンがその入力にあるとすれば、本発明による垂直アンサンブラは高さの2倍の出力スキャンを提供することができる。モードセレクションにより、出力ピクセルバリューを多くの方法で形成することができる。

【1981】C. 7. 2 「ポート」

入力2線式インターフェース

in valid

in accept

in data [7:0]

in lastpel

in lastline

出力2線式インターフェース:

out valid

out accept

out data [9:0]

out last

mode [2:0]

npdata [7:0], upaddr, upsel

[3:0], uprstr, upwstr

ramtest

tdin, tdout, tph0, tckm, tcks

ph0, ph1, notrst0

C. 7. 3 「モード」

入力バスモード [2:0] により選択。

【1982】モードレジスタバリュー1及び7は使用されない。

【1983】上記モードの各々において、出力ピクセルは10ビットバリューとして表示され、バイトとしては表示されない。ラウンディングもしくは打ち切りはこのブロックでは行われぬ。必要に応じて、バリューは同じ範囲を使用するために左にシフトされる。

【1984】C. 7. 3. 1 「モード0:Fifo」
ブロックは単にFIFO記憶装置として作用する。出力ピクセル数は入力ピクセル数と全く同じである。バリューは2だけ左にシフトされる。

【1985】C. 7. 3. 2 「モード2:Repeat」

入力スキャン内のあらゆるラインは高さの2倍の出力スキャンを作るために繰り返される。ここでも、ピクセルバリューは2だけ左にシフトされる。

【1986】A→ABACBDBCCDD

C. 7. 3. 3 「モード4:Lower」

各入力ラインは2つの出力ラインを作り出す。この「下位」モードでは、これら2つのラインの2番目のもの（ディスプレイ上の方）が入力ラインと同じである。ペアの最初のものは現在の入力ラインと前の入力ラインの平均である。前のラインが使用されていない場合の最初の入力ラインの場合、入力ラインが繰り返される。

【1987】これは彩度サンプルが下位輝度サンプルと共に配置される場所を選択すべきである。

【1988】A→ABAC (A+B) / 2 DB (B+C) / 2 C (C+D) / 2 D

C. 7. 3. 4 「モード5:Upper」

「下位」モードと同様であるが、この場合入力ラインは出力ペアの上位を形成し、下位は隣接する入力ラインの平均である。最後の出力ラインは最後の入力ラインの線

り返しである。

【1989】これは彩度サンプルが上位輝度サンプルと共に配置される場所を選択すべきである。

【1990】 $A \rightarrow AB (A+B) / 2$ $CBD (B+C) / 2$ $C (C+D) / 2$ DD

C. 7. 3. 5 「モード6: Central」

この「中央」モードは彩度サンプルが輝度サンプルとの中間にある点に対応する。出力彩度ピクセルを輝度ピクセルと共に配置するために、重み付き平均を使用して出力ラインを形成する。

【1991】 $A \rightarrow AE (3A+B) / 4$ $C (A+B) / 4$ $D (3B+C) / 4$ $(B+3C) / 4$ $(3C+D) / 4$ $(C+3D) / 4$ D

C. 7. 4 「作用の仕方」

想像上aとbで指定される2つのラインストアがある。FIFO及びrepeatモードでは、ラインストアaだけが使用される。各ストアは512ピクセルまでのラインを収容できる(垂直アップサンプリングは水平アップサンプリングの前に行われるべきである)。FIFOモードにはラインの長さに関する制約がない。

【1992】in lastpel及びin last line内の入力信号は、入力ラインの終わりとピクチャの終わりを指示するために使用される。in lastpelにおいては、入力信号は各ラインの最後のピクセルと強く一致すべきである。in lastlineにおいては、入力信号はピクチャの最後のラインの最後のピクセルと強く一致すべきである。

【1993】出力信号out lastは各出力ラインの最後のピクセルと強く一致する。repeatモードでは、各ラインはストアaに書き込まれる。ラインは次に2度読み出される。2度目に読み出される時、次のラインの書き込みをスタートしてもよい。

【1994】lower、upper及びcentralモードでは、ラインはストアa及びbに交互に書き込まれる。ピクチャの最初のラインは常にストアaに書き込まれる。2個の小さなステートマシン、各ストア用に1つずつ、が各々のストアに何があるか、そしてどの出力ラインが形成されているかを記憶する。これらのステートからリードとライトのリクエスト、及び次のラインをいつ現在のデータにオーバーライトするかを決定する信号がラインストアRAMに発せられる。

10 【1995】in lastpelが高い時、レジスタ(lastaddr)がライトアドレスを記憶し、それによって、出力ラインの形成のためにライン長を提供する。

C. 7. 5 「UPI」

このブロックは2個の512x8ビットのRAMアレイを含み、それは典型的な方法でマイクロプロセッサインターフェースを介してアクセスされ得る。マイクロプロセッサアクセスを持つレジスタはない。

【1996】セクションC. 8 「水平アップ・サンブラ」

C. 8. 1 「展望」

本発明においては、トップレベルレジスタは、各色成分のために1つずつ、3個の同じ水平アップ・サンブラを具備する。3個全ては別々に制御されるので、1つだけについてここで説明する。ユーザーの観点から、唯一の違いは各水平アップ・サンブラがメモリーマップ内の異なるアドレスセットにマップされることである。

【1997】水平アップ・サンブラは複製とフィルタリングを組み合わせたオペレーションを遂行する。全てにおいて、4つのオペレーションモードがある。

【1998】

【表264】

モード	機 能
0	直接的(ノープロセッシング)、リセットステート。
1	ノーアップ・サンプリング、フィルタは3タップFIRフィルタを使用する。
2	x2アップ・サンプリング及びフィルタリング
3	x4アップ・サンプリング及びフィルタリング。

表C. 7. 1 水平アップ・サンブラのモード

C. 8. 2 「水平アップ・サンブラの使用」

各水平アップ・サンブラ用のアドレスマップは12個の13ビット係数レジスタと1個の2ビットモードレジスタに対応する25のロケーションから構成される。モードレジスタに書き込まれるナンバーが、表264に概略するように、オペレーションモードを決定する。モードに応じて、一部または全部の係数レジスタを使用でき

る。各係FIRフィルタについて下記に説明する。

【1999】オペレーションモードに応じて、入力、Xnは1、2、または4クロック期間の間一定に保持される。各モードのためにプログラムされる実際の係数は以下の通りである。

【2000】

【表265】

係数	全てのクロック期間
k 0	c 0 0
k 1	c 1 0
k 2	c 2 0

【2001】

【表266】

表C. 7. 2 モード1用の係数

係数	第1のクロック期間	第2のクロック期間
k 0	c 0 0	c 0 1
k 1	c 1 0	c 1 1
k 2	c 2 0	c 2 1

表C. 7. 3 モード1用の係数

【2002】

【表267】

係数	第1の クロック期間	第2の クロック期間	第3の クロック期間	第4の クロック期間
k 0	c 0 0	c 0 1	c 0 2	c 0 3
k 1	c 1 0	c 1 1	c 1 2	c 1 3
k 2	c 2 0	c 2 1	c 2 2	c 2 3

表C. 7. 4 モード1用の係数

特定のモードで使用されない係数は、そのモードで操作する時にはプログラムする必要がない。

【2003】均整のとれたフィルタリングを行うために各ラインの最初と最後のピクセルがフィルタリングの前に繰り返される。例えば、2によってアップ・サンプリングをする場合、各ラインの最初と最後のピクセルが2度ではなく、4度複製される。フィルタ内の残留データが各ラインの終わりで捨てられるので、ピクセル数の出力は常に正確に入カストリーム内の数字の1倍、2倍ま

たは4倍である。

30 【2004】係数値に応じて、出力サンプルは入力サンプルと一致して置かれるか、あるいは入力サンプルからシフトされて置かれる。あるサンプルモードでの係数値の幾つかの例を下記に記す。“-”は係数値が「気にしなくてよい」であることを指示する。全ての値を16進法で示す。

【2005】

【表268】

係数	×2 アップ・サンプル、 i/pと一致する c/pピクセル	×2 アップ・サンプル、 i/p間にある c/pピクセル	×4 アップ・サンプル、 i/p間にある c/pピクセル
c00	0000	01BD	00E9
c01	0000	010B	00B6
c02	.	.	012A
c03	.	.	0102
c10	0800	0538	0661
c11	400	0538	0661
c12	.	.	0446
c13	.	.	029F
c20	0000	010B	00B6
c21	0400	01BD	00E9
c22	.	.	0290
c23	.	.	045F

表C. 7. 5 サンプル係数

C. 8. 3 「水平アップ・サンプラの説明」

水平アップ・サンプラのデータバスを図177に示す。

【2006】オペレーションは×4アップ・サンプルの場合を取り上げ、下記に概略する。それに加えて、×2アップ・サンプリングと×1フィルタリング（モード2及び1）はこれの変質ケースであり、全フィルタを迂回し（モード0）、データは図示するように、入力ラッチから出力ラッチへと最後のマックスを介して直接進む。

【2007】1）有効なデータが入力においてラッチされる場合

ラッチ（L）、それは4クロック期間の間保持される。

【2008】2）（C/OE/FFというラベルが貼られた）係数レジスタは、それぞれ順番に、4個の（P I

PEというラベルが貼られた）パイプラインレジスタの2セットがクロックされるのと同時に、1クロック期間の間乗算器の上に多重送信される。こうして、入力データXnのために、最初のPIPEにバリューc00・Xn、c01・Xn、c02・Xn、c03・Xnが詰められるであろう。

【2009】3）同様に、第2の乗算器が順にその係数にXnを掛け、第3の乗算器が順にその係数全てにXnを掛けるであろう。

【2010】出力は表269に示した形態のものであろうことが解るであろう。

【2011】

【表269】

クロック期間	出力
0	$c20X_n + c10X_{n-1} + c00X_{n-2}$
1	$c21X_n + c11X_{n-1} + c01X_{n-2}$
2	$c22X_n + c12X_{n-1} + c02X_{n-2}$
3	$c23X_n + c13X_{n-1} + c03X_{n-2}$

表C. 7. 6 モード3用の出力シーケンス

出力の観点から、各クロック期間は個々のピクセルを作り出す。各出力ピクセルが12入力ピクセルの重み付きバリューに依存するので（3つしか異なるバリューは存在しないが）、これは×4アップ・サンプルされた入力ピクセル上に12タップフィルタを実装するものとして考えられる。

【2012】×2アップサンプリングにとっても、オペレーションは本質的に同じであるが、入力データが2クロック期間の間だけ保持されることが異なる。更に、2個だけの係数が使用され、PIPEブロックは図示した乗算器によって短くされる。×1フィルタリング用に、

40 入力1クロック期間の間だけ保持される。期待されるように、1つの係数と1つのPIPEステージが使用される。

【2013】次に、本発明における実装の特異性に関して少々説明する。

【2014】1）色空間コンバータ用に設計されたように、データバス幅及び係数幅（13ビットの2の補数）が選ばれ、同じ乗算器を使用することができる。これらの幅は水平アップ・サンプラのためにこれ以上適切なものはない。

50 【2015】2）係数を乗算器上に多重送信するマルチ

ブレイクサは、UPI リードバックと共有される。これは概略図の構造において（基本的にCCODE発生における困難さの故に）ある種の複雑さを導き出すが、実際の回路はそれより小さなものである。

【2016】3）色空間コンバータにおけるように、キャリセブ乗算器が使用され、その結果は終わりの時点まで分解されるだけである。

【2017】水平アップ・サンブラ全体用の制御は、その入力上にある時、入力におけるデータ量の2倍もしくは4倍の量を作り出すことができる1つの2線式インターフェースステージと見なすことができる。UPIを介してプログラムされるモードが、プログラム可能シフトレジスタ（bob）の長さを決定する。選択されるモードはクロック期間毎、2クロック期間毎、あるいは4クロック期間毎に出力パルスを生み出す。次にこれは主ステートマシンを制御するが、そのステートは更に（2線式インターフェース用に）invalid、out accept、及び信号in lastによって決定される。この信号は垂直アップ・サンブラから送られ、各ラインの最後のピクセル用に高くなる。これは各ラインの最初と最後のピクセルが2度複製されるようにし、（ラインが完了した直後に、パイプラインが部分的に処理された冗長データを含む）ライン間のパイプラインをクリアする。

【2018】セクションC. 9 「色空間コンバータ」
C. 9. 1 「展望」

本発明における色空間コンバータ（CSC）は入ってくる9ビットデータに3×3マトリックスのかけ算を行い、その後加算を行う：

【2019】

【数6】

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} c_{01} & c_{02} & c_{03} \\ c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} c_{04} \\ c_{14} \\ c_{24} \end{bmatrix}$$

30

x0-2が入力データである場合、y0-2が出力データとなり、c_{nm}が係数である。概略図において名前は信号名に対応するので、マトリックス係数に関して、少々型破りのネーミングは慎重にすべきである。

【2020】CSCは多くの異なる色空間の間の変換を実行することができるが、これらの変換の限定されたセットがトップレベルレジスタにおいて使用される。デザイン色空間変換は以下の通りである：

ER, EG, EB → Y, CR, CB
R, G, B → Y, CR, CB
Y, CR, CB → ER, EG, EB
Y, CR, CB → R, G, B

式中、R、G及びBは範囲（0..511）内にあり、他の全ての量は（32..470）の範囲にある。トップレベルレジスタCSCに対する入力、Y、CR、CBであるので、これらの式の3番目と4番目の式だけが関連する。

【2021】CSCデザインでは、係数の精度は、9ビットデータ用には、全ての出力がアルゴリズムの全浮動小数点シミュレーションにより作られるバリューの+1または-1ビット以内であるように選ばれた（これは達成し得る最良の精度である）。これはcx0-cx3用に13ビットの2の補数係数を与え、cx4用に14ビットの2の補数係数を与えた。全てのデザイン変換用の係数を下記に、10進法と6進法の両方で記す。

【2022】

【表270】

$$E_R, E_G, E_B \rightarrow Y, C_R, C_B$$

$$R, G, B \rightarrow Y, C_R, C_B$$

$$Y, C_R, C_B \rightarrow E_R, E_G, E_B$$

$$Y, C_R, C_B \rightarrow R, G, B$$

係数	Er→Y		R→Y	
	10進法	6進法	10進法	6進法
c01	0.299	0132	0.256	
c02	0.587	0259	0.502	
c03	0.114	0075	0.098	
c04	0.0	0000	16	
c11	0.5	0200	0.428	
c12	-0.419	FE53	-0.358	
c13	-0.081	FFAD	-0.070	
c14	128.0	0800	128	
c21	-0.169	FF53	-0.144	
c22	-0.331	FEAD	-0.283	
c23	0.0	0200	0.428	
c24	128	0800	128	

表C. 8. 1 様々な変換用の係数 (1/2)

[2023]

【表271】

係数	Er→Y		R→Y	
	10進法	6進法	10進法	6進法
c01	1.0	0400	1.159	04AD
c02	1.402	059C	1.538	C66E
c03	0.0	0000	0.0	0000
c04	-179.456	F4C8	-223.478	F1B3
c11	1.0	04C0	1.188	C4AD
c12	-0.714	FD25	-0.335	FCA9
c13	-0.344	FEAD	-0.402	FE64
c14	135.5	0878	139.7	0BBA
c21	1.0	0400	1.159	04AD
c22	0.0	0000	0.0	0000
c23	1.772	0717	2.071	CB49
c24	-226.816	F1D2	-283.34	EE42

表C. 8. 1 様々な変換用の係数 (2/2)

これら全てのナンバーは以下の基本式：

$$Y = 0.299ER + 0.587EG + 0.0114EB$$

そして、以下の色差式から計算される：

$$CR = ER - Y$$

$$CB = EB - Y$$

R、G及びBにおける式はこれらの量の全範囲を考慮した後、これらから引き出される。

【2024】C. 9. 2 「色空間コンバータの使用」
リセット後、c01、c12、c23が1に設定され、他の全ての係数が0に設定される。こうして、 $y0 = x0$ 、 $y1 = x1$ 、 $y2 = x2$ 、及び全てのデータが無変更のまま送られる。色空間変換を選択するには、単に（例えば、表270、表271からの）適当な係数をアドレスマップ内に指定されるロケーションに書き込むだけでよい。

【2025】概略図において、 $x0..2$ はin data 0..2に対応し、 $y0..2$ はout data 0..2に対応する。ユーザーはCSCに対する入力データを4:4:4にアップ・サンプルしなければならないことを覚えておくべきである。これを怠ると、色空間変換が何の意味も持たなくなるばかりでなく、チップもロックするであろう。

【2026】注意すべきことは、各々の出力が係数と入カプラス（またはマイナス）定数の可能な組み合わせから形成され得ることである。このように、所定の色空間変換のために、出力オーダーを変換マトリックス（係数）が書き込まれるアドレス）内のローをスワッピングすることにより変更できる。

【2027】表270、表271における全ての変換のために、CSCが作動することは保証されている。他の変換を用いる場合、ユーザーは以下の事項を覚えておかねばならない：

1) 計算の中間結果が（サインビットを除き）10ビット以上の精度を要求する場合、ハードウェアは作動しないであろう。

【2028】2) CSCの出力は0と511にサチュレートされる。つまり、0以下の数が0で置き換えられ、511以上の数が511で置き換えられる。サチュレーションロジックの実行は、結果がわずかに511を越えているか、わずかに0以下であると仮定する。CSCが不正確にプログラムされた場合、出力が常に（あるいはほとんどの場合）サチュレートするよう見えることが共通のきざしであろう。

【2029】C. 9. 3 「CSCの説明」

CSCの構造を図178に図示するが、スペース上の制約から、3つの「コンポーネント」の内2つだけを示した。図中、「レジスタ」もしくはRはマスター・スレーブレジスタを示し、「ラッチ」もしくはLは透明ラッチを示す。

【2030】全ての係数は図において明示されていないリード・ライトUPIレジスタにロードされる。オペレーションを理解するために、左端の「コンポーネント」(出力out_data0を作り出す)に関して、以下のシーケンスを考慮する：

1) データは入力x0-2 (in_data0-2) に到達する。これは入力色空間における1つのピクセルを表す。これがラッチされる。

【2031】2) x0にc01を掛け、最初のパイプラインレジスタにラッチする。x1及びx2が1つのレジスタ上を移動する。

【2032】3) x1にc02を掛け、それを(x1, c01)に加算し、次のパイプラインレジスタにラッチする。x2が1つのレジスタ上を移動する。

【2033】4) x2にc03を掛け、それを(3)の結果に加算し、(x1, c01+x2, c02+x3, c03)を生み出す。その結果を次のパイプラインレジスタにラッチする。

【2034】5) (4)の結果をc04に加算する。データは乗算器を通してキャリセーブフォーマットの中に保持されているので、このアダプターも乗算器チェーンからのデータを分解するために使用される。その結果を次のパイプラインレジスタにラッチする。

【2035】6) 最終的なオペレーションはデータをサチュレートすることである。これを行うために、部分的結果が分解アダプターからサチュレートブロックに送られる。

【2036】このセクションの開始時点のマトリックス式において明記したように、その結果がy0であることが解る。同様に、y1とy2が同じ方法で形成される。

【2037】係数を被乗数とし、データを乗数として、3個の乗算器を使用した。これにより効率的なレイアウトを達成でき、部分的結果がデータバスを下り、同じ入力データが各出力に1つずつの3つの並列する同じデータバスを横切って送られた。セクションC. 9. 2において説明したりセットステートを達成するために、3つの「コンポーネント」の各々を異なる方法でリセットしなければならない。3セットの概略図と3つのわずかに異なるレイアウトを持つことを避けるため、これはトップレベルで高または低につながるUPIレジスタに対する入力を持つことにより達成される。

【2038】CSCはそれに関連する制御をほとんど持たない。にもかかわらず、各々のパイプラインステージは2線式インターフェースステージであるので、それら

t_r+lin_validr)を持つ有効ラッチ及びアクセプトラッチのチェーンがある。従って、CSCは5ステージの深さの2線式インターフェースであり、ストールされた場合10レベルのデータを保持することができる。

【2039】出力パイプに於ける次の機能が異なるクロック発生器から外れるので、CSCの出力は再同期化ラッチを含む。

【2040】セクション C. 10 「出力コントローラ」

C. 10. 1 「序文」

本発明による出力コントローラは以下の機能を果たす：

- ・ 3モードの1つにおいてデータを提供する
- ・ 24ビット 4:4:4
- ・ 16ビット 4:2:2
- ・ 8ビット 4:2:2
- ・ vsyncとhsyncパルスにより、またプログラムされたタイミングレジスタにより定義されるビデオ表示ウィンドーにデータを整列させる
- ・ 必要に応じて、ビデオウィンドーのまわりのボーダーを加える

C. 10. 2 「ポート」

入力2線式インターフェース：

- ・ in_valid
- ・ in_accept
- ・ in_data[23:0]

出力2線式インターフェース：

- ・ out_valid
- ・ out_accept
- ・ out_data[23:0]
- ・ out_active
- ・ out_window
- ・ out_comp[1:0]

in_vsync、in_hsync
nupdata[7:0]、upaddr[4:0]、
upsel、rstr、wstr、tdin、tdout、
tph0、tckm、tcks_chiptes
t、ph0、ph1、notrst0、notrst1
C. 10. 3 「アウトモード」

出力のフォーマットはopmodeレジスタに書き込むことにより、選択される。

【2041】C. 10. 3. 1 「モード0」

このモードは24ビットの4:4:4RGBもしくはYCrCbである。入力データは直接出力に進む。

【2042】C. 10. 3. 2 「モード1及び2」

これらのモードは4:2:2のYCrCbを表す。In_data[23:16]がYであると仮定すると、in_data[15:8]はCrであり、in_data[7:0]はCbである。

【2043】C. 10. 3. 2. 1 「モード1」

621

16ビットのYCrCbにおいて、Yはout data [15:8] 上に表示される。CrとCbはout data [7:0] 上で、Cbを最初にして、時分割多重化される。Out data [23:16] は使用されない。

【2044】C. 10. 3. 2. 2 「モード2」

8ビットのYCrCbにおいて、Y、Cr及びCbはCb、Y、Cr、Yの順序で、out data [7:0] 上で時分割多重化される。Out data [23:16] は使用されない。

【2045】C. 10. 3. 3 「出力タイミング」

ビデオディスプレイウインドーにデータを置くために、以下のレジスタが使用される。

【2046】・Vdelay-ビデオまたはボーダーの最初のラインの前に、vsyncパルスに続くhsyncパルス数

・hdelay-hsyncと、ビデオまたはボーダーの最初のピクセルとの間のクロックサイクル数

・height-ラインにおけるビデオウインドーの高さ

・width-ピクセルにおけるビデオウインドーの幅

・north, south-ラインにおけるビデオウインドーの上及び下の各々のボーダーの高さ、

・west, east-ピクセルにおけるビデオウインドーの左及び右に対する

各々のボーダー幅

最小のvdelayは0である。最初のhsyncは最初の活動的ラインである。hdelayにプログラムされ得る最小値は2である。しかしながら、inhsyncから最初の活動的出力ピクセルまでの実際のディレイはhdelay+1サイクルであることに注意。

【2047】ボーダーの縁はバリュー0を持つことができる。ボーダーの色はレジスタにborder r、border g、及びborder bを書くことによって選択される。ボーダーの外側のエリアの色はレジスタにblank r、blank g、blank bを書くことによって選択される。出力モード1及び2において遂行されるマルチプレクシングもボーダー及びブランクコンポーネントに影響を与えるであろう。つまり、これらのレジスタのバリューはin data [23:16]、in data [15:8]、及びin data [7:0] に対応する。

【2048】C. 10. 4 「出力フラグ」

・out activeは出力データが活性ウインドー、つまりビデオデータまたはボーダーの一部であることを指示する。

【2049】・out windowは出力データがビデオウインドーの一部であることを指示する。

【2050】・out comp [1:0] は色成分が出力モード1及び2のout data [7:0] の上

622

に存在することを指示する。モード1においては、0=Cb、1=Crである。モード2においては、0=Y、1=Cr、2=Cbである。

C. 10. 5 「2線式モード」

本発明の2線式モードは2線式レジスタに1を書き込むことによって選択される。それはリセットに引き続いては選択されない。2線式モードでは、出力タイミングレジスタとsync信号が無視され、ブロックを通るデータフローはout acceptによって制御される。

10 通常のオペレーションでは、out acceptは両方につながれるべきであることに注意。

【2051】C. 10. 6 「スノーパ」

出力フラグへのアクセスを含むブロックの出力上にスノーパスノーパがある。

C. 10. 7 「作用の仕方」

2個の同じダウンカウンタがディスプレイにおける現在の位置を覚えている。Vcountはhsyncs上で減少し、vsync上の、あるいはその終末カウントにおける適当なタイミングレジスタからロードする。Hcountはあらゆるピクセル上で減少し、hsync上で、あるいはその終末カウントにおいてロードする。出力モード2では、1つのピクセルが2個のクロックサイクルに対応することに注意。

【2052】セクション C. 11 「クロック分周器」

C. 11. 1 「展望」

本発明におけるトップレベルレジスタは、1つはPICTURE CLKを発生させるため、1つはAUDIO CLKを発生させるために、2個の同じクロック分周器を含む。クロック分周器は同じであり、別々に制御される。従って、1つだけをここで取り上げて説明する。ユーザーの観点から、唯一の違いは各クロック分周器の除数レジスタがメモリーマップ内のアドレスの異なるセットにマップされることである。

【2053】クロック分周器の機能は4Xsysclk分配クロック周波数を提供することであり、偶数のマーク対スペース比のための要件はない。

【2054】除数は~0から~16,000,000の範囲にあることが求められるので、最小除数が16であるという制約付きで24ビットを用いて表すことができる。これはクロック分周器が除数/2を用いて、(1つのsysclkサイクル内に)等しいマーク対スペース比を接近させるであろうからである。利用できる最大クロック周波数はsysclkであるので、利用できる最大分配周波数はsysclk/2である。更に、4個のカウンタが連続で使用されるので、除数/2は決して8以下であってはならず、そうでなければ分配されるクロック出力は正のパワーレールに駆動されるであろう。

【2055】C. 11. 2 「クロック分周器の使用」

各クロック分周器用のアドレスマップは3個の8ビット

除数レジスタと1個の1ビットアクセスレジスタに対応する4つのロケーションから構成される。クロック分周器は無活動のものをパワーアップし、その除数レジスタに対するアクセス完了によって活性化される。

【2056】除数レジスタは表272のアドレスマップに従ったオーダーで書き込まれてよい。クロック分周器はそのアクセスビットにおける同期化0~1遷移を感じることにより活性化される。遷移が初めて感知されると、クロック分周器はリセット状態から出て、分配されたクロックを発生させる。(除数も変更されていると仮定して)それに続く遷移は、クロック分周器を単にその新しい周波数on-the-flyにロックさせるであろう。一度活性化されると、クロック分周器を停止させる方法はチップRESET以外にない。

【2057】

【表272】

アドレス	レジスタ
00b	アクセスビット
01b	除数MSB
10b	除数
11b	除数LSB

表C. 10. 1 クロック分配器レジスタ

16~16、777、216の範囲の除数値が使用できる。

【2058】C. 11. 3 「クロック分周器の説明」
クロック分周器は、1カウンタが桁上げするにつれて、それが次のカウンタを順に活性化させるように縦続される4個の22ビットカウンタとして実装される。カウンタは桁上げの前に除数/4のバリュをカウントダウンするので、各カウンタがそれを順に取り上げ、分配クロック周波数のパルスを発生させるであろう。

【2059】桁上げの後、カウンタは除数/8を再びロードし、これがカウントダウンされてほぼ等しいマーク対スペース比分配クロックを作り出す。各々のカウンタが前のカウンタにより活性化される時、除数レジスタから再ロードするにつれて、これは単に除数の内容を変更

することによって、分配クロック周波数をオン・ザ・フライで変更することができるようにする。

【2060】各カウンタはカウンタ間のクロックスキューを正確に制御し、各カウンタが別個のクロックセットによってクロックされ得るようにするため、それ自身の独立したクロック発生器によってクロックされる。

【2061】ステートマシンは除数/4及び除数/8バリュの発生器を制御し、更にPLLからクロック発生器へと正しいソースクロックを多重送信する。カウンタは除数のバリュに応じて異なるクロックによりクロックされる。これは異なる除数バリュが、PLLから提供されるクロックの異なる組合せを用いてそのエッジが配置される分配クロックを作り出すであろうからである。

【2062】C. 11. 4 クロック分周器のテスト
CHIPTEST Highでチップをパワーアップすることによりクロック分周器をテストすることができる。これはPLLによって発生されるクロックに対抗するものとして、sysclkによりクロックされるクロック分周器におけるクロック化ロジックの全てを強制する効果を持つ。

【2063】クロック分周器はフルスキャンを持つように設計されているので、チップが上記のように動かされる限り、標準のJTAGアクセスを用いて引続きテストされ得る。

【2064】装置が通常のオペレーションで動いている一方、CHIPTESTがHighに保持される場合、クロック分周器の機能性は保証されない。

【2065】セクション C. 12 「アドレスマップ」

C. 12. 1 「トップレベルアドレスマップ」

注：-

1) 表273~表281に記載するトップレベルアドレスマップ用のレジスタはデザインの中で使用される名前である。それらは必ずしもデータシートに現れる名前でもなくてもよい。

【2066】2) これはフルアドレスマップであるので、ここにリストアップするロケーションの多くはテスト用のみのロケーションを含む。

【2067】

【表273】

レジスタ名	アドレス	ビット	コメント
BU EVENT	0x0	8	リセットするために1を書き込む
BU MASK	0x1	8	R/W
BU EN INTERRUPTS	0x2	1	R/W
BU WADDR COD STD	0x4	2	R/W
BU WADDR ACCESS	0x5	1	R/Wアクセス
BU WADDR CTL0	0x6	3	R/W
BU DISP ADDR LINES IN LAST ROW0	0x8	3	R/W
BU DISP ADDR LINES IN LAST ROW1	0x9	3	R/W
BU DISP ADDR LINES IN LAST ROW2	0xa	3	R/W
BU DISPADDR ACCESS	0xb	1	R/Wアクセス
BU DISPADDR CTL0	0xc	8	R/W
BU DISPADDR CTL1	0xd	1	R/W

表C.11.1

トップレベルレジスタA、トップレベルアドレスマップ (1/9)

[2068]

[表274]

レジスタ名	アドレス	ビット	コメント
BU BM ACCESS	0x10	1	R/Wアクセス
BU BM CTL0	0x11	2	R/W
BU BM TARGET IX	0x12	4	R/W
BU BM PRES NUM	0x13	8	R/W非同期的
BU BM THIS PNUM	0x14	8	R/W
BU BM PIC NUM0	0x15	8	R/W
BU BM PIC NUM1	0x16	8	R/W
BU BM PIC NUM2	0x17	8	R/W
BU BM TEMP REF	0x18	5	R/O
BU ADDRGEN KEYHOLE ADDR MSB	0x28	1	R/Wアドレス発生 書き込みエラー発生 内容については表C.11.2を参照
BU ADDRGEN KEYHOLE ADDR LSB	0x29	8	
BU ADDRGEN KEYHOLE DATA	0x2a	8	
BU IT PAGE START	0x30	5	R/W
BU IT READ CYCLE	0x31	4	R/W
BU IT WRITE CYCLE	0x32	4	R/W

表C.11.1

トップレベルレジスタA、トップレベルアドレスマップ (2/9)

[2069]

[表275]

レジスタ名	アドレス	ビット	コメント
BU IT REFRESH CYCLE	0x33	4	R/W
BU IT RAS FALLING	0x34	4	R/W
BU IT CAS FALLING	0x35	4	R/W
BU IT CONFIG	0x36	1	R/W
BU OC ACCESS	0x40	1	R/Wアクセス
BU OC MODE	0x41	2	R/W
BU OC 2WIRE	0x42	1	R/W
BU OC BORDER R	0x49	8	R/W
BU OC BORDER G	0x4a	8	R/W
BU OC BORDER B	0x4b	8	R/W
BU OC BLANK R	0x4d	8	R/W
BU OC BLANK G	0x4e	8	R/W
BU OC BLANK B	0x4f	8	R/W
BU OC HDELAY 1	0x50	3	R/W
BU OC HDELAY 0	0x51	8	R/W
BU OC WEST 1	0x52	3	R/W
BU OC WEST 0	0x53	8	R/W

表C. 11. 1

トップレベルレジスタA、トップレベルアドレスマップ (3/9)

[2070]

【表276】

レジスタ名	アドレス	ビット	コメント
BU OC EAST 1	0x54	3	R/W
BU OC EAST 0	0x55	8	R/W
BU OC WIDTH 1	0x56	3	R/W
BU OC WIDTH 0	0x57	8	R/W
BU OC VDELAY 1	0x58	3	R/W
BU OC VDELAY 0	0x59	8	R/W
BU OC NORTH 1	0x5a	3	R/W
BU OC NORTH 0	0x5b	8	R/W
BU OC SOUTH 1	0x5c	3	R/W
BU OC SOUTH 0	0x5d	8	R/W
BU OC HEIGHT 1	0x5e	3	R/W
BU OC HEIGHT 0	0x5f	8	R/W
BU IF CONFIGURE	0x60	5	R/W
BU UV MODE	0x61	6	R/W, xnnnnnnnn
BU COEFF KEYADDR	0x62	7	R/W 内容について は表C. 11. 3 を参照
BU COEFF KEYDATA	0x63	8	
BU GA ACCESS	0x68	1	R/W
BU GA BYPASS	0x69	1	R/W

表C. 11. 1

トップレベルレジスタA、トップレベルアドレスマップ (4/8)

[2071]

【表277】

レジスタ名	アドレス	ビット	コメント
BU GA RAM0 ADDR	0x6a	8	R/W
BU GA RAM0 DATA	0x6b	8	R/W
BU GA RAM1 ADDR	0x6c	8	R/W
BU GA RAM1 DATA	0x6d	8	R/W
BU GA RAM2 ADDR	0x6e	8	R/W
BU GA RAM2 DATA	0x6f	8	R/W
BU DIVA 3	0x70	1	R/W
BU DIVA 2	0x71	8	R/W
BU DIVA 1	0x72	8	R/W
BU DIVA 0	0x73	8	R/W
BU DIVP 3	0x74	1	R/W
BU DIVP 2	0x75	8	R/W
BU DIVP 1	0x76	8	R/W
BU DIVP 0	0x77	8	R/W
BU PAD CONFIG 1	0x78	7	R/W
BU PAD CONFIG 0	0x79	8	R/W
BU PULL RESISTORS	0x7a	8	R/W
BU REF INTERVAL	0x7b	8	R/W
BU REVISION	0xff	8	RO-修正

表C. 11. 1

トップレベルレジスタA、トップレベルアドレスマップ (5/9)

[2072]

[表278]

レジスタ名	アドレス	ビット	コメント
以下のレジスタは「テスト空間」にある。			
それらはデータシートには現れないであろう。			
BU BM PRES FLAG	0x80	1	R/W
BU BM EXP. TR	0x81	**	これらのレジスタは メモリ上では省略 する。
BU BM TR DELTA	0x82	**	
BU BM ARR IX	0x83	2	R/W
BU BM DSP IX	0x84	2	R/W
BU BM RDY IX	0x85	2	R/W
BU BM BSTATE3	0x86	2	R/W
BU BM BSTATE2	0x87	2	R/W
BU BM BSTATE1	0x88	2	R/W
BU BM INDEX	0x89	2	R/W
BU BM STATE	0x8a	5	R/W
BU BM FROMPS	0x8b	1	R/W
BU BM FROMFL	0x8c	1	R/W

表C. 11. 1

トップレベルレジスタA、トップレベルアドレスマップ (6/9)

[2073]

[表279]

レジスタ名	アドレス	ビット	コメント
BU DA CCMP0 SNP3	0x80	8	R/W、これらは 専断アドレス発生器 のスケールである
BU DA CCMP0 SNP2	0x91	8	
BU DA CCMP0 SNP1	0x82	8	
BU DA CCMP0 SNP0	0x93	8	
BU DA CCMP1 SNP3	0x80	8	
BU DA CCMP1 SNP2	0x91	8	
BU DA CCMP1 SNP1	0x82	8	
BU DA CCMP1 SNP0	0x93	8	
BU DA CCMP2 SNP3	0x80	8	
BU DA CCMP2 SNP2	0x8a	8	
BU DA CCMP2 SNP1	0x8b	8	
BU DA CCMP2 SNP0	0xa0	8	

表C.11.1

トップレベルレジスタA、トップレベルアドレスマップ (7/8)

[2074]

[表280]

レジスタ名	アドレス	ビット	コメント
BU UV RAM1A ADDR 1	0xa0	8	R/W、垂直アップ サンブラのRAMに アクセス
BU UV RAM1A ADDR 0	0xa1	8	
BU UV RAM1A DATA	0xa2	8	
BU UV RAM1B ADDR 1	0xa4	8	
BU UV RAM1B ADDR 0	0xa5	8	
BU UV RAM1B DATA	0xa6	8	
BU UV RAM2A ADDR 1	0xa8	8	
BU UV RAM2A ADDR 0	0xa9	8	
BU UV RAM2A DATA	0xaa	8	
BU UV RAM2B ADDR 1	0xac	8	
BU UV RAM2B ADDR 0	0xad	8	
BU UV RAM2B DATA	0xae	8	

表C.11.1

トップレベルレジスタA、トップレベルアドレスマップ (8/9)

[2075]

[表281]

レジスタ名	アドレス	ビット	コメント
BU WA ADDR SNP2	0xb0	8	R/W アドレス発生 キーホルスペース
BU WA ADDR SNP1	0xb1	8	
BU WA ADDR SNP0	0xb2	8	
BU WA DATA SNP1	0xb4	8	R/W データの出力上
BU WA DATA SNP0	0xb5	8	
BU IF SNP0 1	0xb8	8	R/W 出力データも スキャン
BU IF SNP0 0	0xb9	8	
BU IF SNP1 1	0xba	8	
BU IF SNP1 0	0xbb	8	
BU IF SNP2 1	0xbc	8	
BU IF SNP2 0	0xbd	8	R/W RAMへの アクセス
BU IFRAM ADDR 1	0xc0	1	
BU IFRAM ADDR 0	0xc1	8	
BU IFRAM DATA	0xc2	8	
BU OC SNP 3	0xc4	8	チップの出力上
BU OC SNP 2	0xc5	8	
BU OC SNP 1	0xc6	8	
BU OC SNP 0	0xc7	8	
BU YAPLL CONFIG	0xc8	8	R/W
BU BM FRONT	0xca	1	R/W
BYPASS			

表C. 11. 1

トップレベルレジスタA、トップレベルアドレスマップ (9/9)

C. 12. 1 「アドレス発生器キーホールスペース」
アドレス発生器キーホール表に関する注：

1) アドレス発生器キーホール内の全てのレジスタは、
それらの幅に関わらず、4バイトのアドレススペースを
取る。省略アドレス (0x00、0x04等) は常に0
を読み返す。

【2076】2) 関連ブロック (dispaddr また
はwaddrgen) のアクセスビットはこのキーホール
にアクセスする前に設定されなければならない。

【2077】

【表282】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU DISPADDR BUFFER0 BASE MSB	0x01	2	18ビット レジスタ ロードしな ければなら ない
BU DISPADDR BUFFER0 BASE MID	0x02	8	
BU DISPADDR BUFFER0 BASE LSB	0x03	8	
BU DISPADDR BUFFER1 BASE MSB	0x05	2	ロードしな ければなら ない
BU DISPADDR BUFFER1 BASE MID	0x06	8	
BU DISPADDR BUFFER1 BASE LSB	0x07	8	
BU DISPADDR BUFFER2 BASE MSB	0x09	2	ロードしな ければなら ない
BU DISPADDR BUFFER2 BASE MID	0x0a	8	
BU DISPADDR BUFFER2 BASE LSB	0x0b	8	
BU OLDPATH LINE0 MSB	0x0d	2	テストのみ
BU OLDPATH LINE0 MID	0x0e	8	
BU OLDPATH LINE0 LSB	0x0f	8	

表C. 11. 2

トップレベルレジスタA、アドレス発生器キーホール (1/19)

[2078]

[表283]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU OLDPATH LINE1 MSB	0x11	2	テストのみ
BU OLDPATH LINE1 MID	0x12	8	
BU OLDPATH LINE1 LSB	0x13	8	
BU OLDPATH LINE2 MSB	0x15	2	テストのみ
BU OLDPATH LINE2 MID	0x16	8	
BU OLDPATH LINE2 LSB	0x17	8	
BU OLDPATH VBCNT0 MSB	0x18	2	テストのみ
BU OLDPATH VBCNT0 MID	0x1a	8	
BU OLDPATH VBCNT0 LSB	0x1b	8	
BU OLDPATH VBCNT1 MSB	0x1c	2	テストのみ
BU OLDPATH VBCNT1 MID	0x1d	8	
BU OLDPATH VBCNT1 LSB	0x1e	8	

表C.11.2

トップレベルレジスタA、アドレス発生器キーホール (2/19)

【2079】

【表284】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU OLDPATH VBCNT2 MSB	0x21	2	テストのみ
BU OLDPATH VBCNT2 MID	0x22	8	
BU OLDPATH VBCNT2 LSB	0x23	8	
BU DISPADDR COMP0 OFFSET MSB	0x25	2	ロードしな ければなら ない
BU DISPADDR COMP0 OFFSET MID	0x26	8	
BU DISPADDR COMP0 OFFSET LSB	0x27	8	
BU DISPADDR COMP1 OFFSET MSB	0x29	2	ロードしな ければなら ない
BU DISPADDR COMP1 OFFSET MID	0x2a	8	
BU DISPADDR COMP1 OFFSET LSB	0x2b	8	
BU DISPADDR COMP2 OFFSET MSB	0x2d	2	ロードしな ければなら ない
BU DISPADDR COMP2 OFFSET MID	0x2e	8	
BU DISPADDR COMP2 OFFSET LSB	0x2f	8	

表C. 11. 2

トップレベルレジスタA、アドレス発生器キーホール (3/19)

【2080】

【表285】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU DISPADDR COMP0 VBS MSB	0x31	2	ロードしな ければなら ない
BU DISPADDR COMP0 VBS MID	0x32	8	
BU DISPADDR COMP0 VBS LSB	0x33	8	
BU DISPADDR COMP1 VBS MSB	0x35	2	ロードしな ければなら ない
BU DISPADDR COMP1 VBS MID	0x36	8	
BU DISPADDR COMP1 VBS LSB	0x37	8	
BU DISPADDR COMP2 VBS MSB	0x39	2	ロードしな ければなら ない
BU DISPADDR COMP2 VBS MID	0x3a	8	
BU DISPADDR COMP2 VBS LSB	0x3b	8	
BU ADDR COMP0 HBS MSB	0x3d	2	ロードしな ければなら ない
BU ADDR COMP0 HBS MID	0x3e	8	
BU ADDR COMP0 HBS LSB	0x3f	8	

表C. 11. 2

トップレベルレジスタA、アドレス発生器キーホール (4/19)

[2081]

[表286]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU ADDR COMP1 HBS MSB	0x41	2	ロードしな ければなら ない
BU ADDR COMP1 HBS MID	0x42	8	
BU ADDR COMP1 HBS LSB	0x43	8	
BU ADDR COMP2 HBS MSB	0x45	2	ロードしな ければなら ない
BU ADDR COMP2 HBS MID	0x46	8	
BU ADDR COMP2 HBS LSB	0x47	8	
BU DISPADDR COMP0 HBS MSB	0x49	2	ロードしな ければなら ない
BU DISPADDR COMP0 HBS MID	0x4a	8	
BU DISPADDR COMP0 HBS LSB	0x4b	8	
BU DISPADDR COMP1 HBS MSB	0x4d	2	ロードしな ければなら ない
BU DISPADDR COMP1 HBS MID	0x4e	8	
BU DISPADDR COMP1 HBS LSB	0x4f	8	

表C. 11. 2

トップレベルレジスタA、アドレス発生器キーホール (5/19)

【2082】

【表287】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU DISPADDR COMP2 HBS MSB	0x51	2	ロードしな ければなら ない
BU DISPADDR COMP2 HBS MID	0x52	8	
BU DISPADDR COMP2 HBS LSB	0x53	8	
BU DISPADDR CNT LEFT0 MSB	0x55	2	
BU DISPADDR CNT LEFT0 MID	0x56	8	テストのみ
BU DISPADDR CNT LEFT0 LSB	0x57	8	
BU DISPADDR CNT LEFT1 MSB	0x59	2	
BU DISPADDR CNT LEFT1 MID	0x5a	8	
BU DISPADDR CNT LEFT1 LSB	0x5b	8	テストのみ
BU DISPADDR CNT LEFT2 MSB	0x5d	2	
BU DISPADDR CNT LEFT2 MID	0x5e	8	
BU DISPADDR CNT LEFT2 LSB	0x5f	8	

表C. 11. 2

トップレベルレジスタA、アドレス発生器キーホール (6/19)

【2083】

【表288】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU DISPADDR PAGE ADDR0 MSB	0x61	2	テストのみ
BU DISPADDR PAGE ADDR0 MID	0x62	8	
BU DISPADDR PAGE ADDR0 LSB	0x63	8	
BU DISPADDR PAGE ADDR1 MSB	0x65	2	テストのみ
BU DISPADDR PAGE ADDR1 MID	0x66	8	
BU DISPADDR PAGE ADDR1 LSB	0x67	8	
BU DISPADDR PAGE ADDR2 MSB	0x69	2	テストのみ
BU DISPADDR PAGE ADDR2 MID	0x6a	8	
BU DISPADDR PAGE ADDR2 LSB	0x6b	8	
BU DISPADDR BLOCK ADDR0 MSB	0x6d	2	テストのみ
BU DISPADDR BLOCK ADDR0 MID	0x6e	8	
BU DISPADDR BLOCK ADDR0 LSB	0x6f	8	

表C.11.2

トップレベルレジスタA、アドレス発生器キーホール (7/18)

[2084]

[表289]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU DISPADDR BLOCK ADDR1 MSB	0x71	2	テストのみ
BU DISPADDR BLOCK ADDR1 MID	0x72	8	
BU DISPADDR BLOCK ADDR1 LSB	0x73	8	
BU DISPADDR BLOCK ADDR2 MSB	0x75	2	テストのみ
BU DISPADDR BLOCK ADDR2 MID	0x76	8	
BU DISPADDR BLOCK ADDR2 LSB	0x77	8	
BU DISPADDR BLOCKS LEFT0 MSB	0x79	2	テストのみ
BU DISPADDR BLOCKS LEFT0 MID	0x7a	8	
BU DISPADDR BLOCKS LEFT0 LSB	0x7b	8	
BU DISPADDR BLOCKS LEFT1 MSB	0x7d	2	テストのみ
BU DISPADDR BLOCKS LEFT1 MID	0x7e	8	
BU DISPADDR BLOCKS LEFT1 LSB	0x7f	8	

表C.11.2

トップレベルレジスタA、アドレス発生器キーホール (8/18)

[2085]

50 [表290]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU DISPADDR BLOCKS LEFT2 MSB	0x81	2	テストのみ
BU DISPADDR BLOCKS LEFT2 MID	0x82	8	
BU DISPADDR BLOCKS LEFT2 LSB	0x83	8	
BU WADDR BUFFER0 BASE MSB	0x85	2	ロードしな ければなら ない
BU WADDR BUFFER0 BASE MID	0x86	8	
BU WADDR BUFFER0 BASE LSB	0x87	8	
BU WADDR BUFFER1 BASE MSB	0x89	2	ロードしな ければなら ない
BU WADDR BUFFER1 BASE MID	0x8a	8	
BU WADDR BUFFER1 BASE LSB	0x8b	8	
BU WADDR BUFFER2 BASE MSB	0x8d	2	ロードしな ければなら ない
BU WADDR BUFFER2 BASE MID	0x8e	8	
BU WADDR BUFFER2 BASE LSB	0x8f	8	

表C. 11. 2

トップレベルレジスタA. アドレス発生器キーホール (9/19)

[2086]

[表291]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP0 HMBADDR MSB	0x91	2	テストのみ
BU WADDR COMP0 HMBADDR MID	0x92	8	
BU WADDR COMP0 HMBADDR LSB	0x93	8	
BU WADDR COMP1 HMBADDR MSB	0x95	2	テストのみ
BU WADDR COMP1 HMBADDR MID	0x96	8	
BU WADDR COMP1 HMBADDR LSB	0x97	8	
BU WADDR COMP2 HMBADDR MSB	0x99	2	テストのみ
BU WADDR COMP2 HMBADDR MID	0x9a	8	
BU WADDR COMP2 HMBADDR LSB	0x9b	8	
BU WADDR COMP0 VMBADDR MSB	0x9d	2	テストのみ
BU WADDR COMP0 VMBADDR MID	0x9e	8	
BU WADDR COMP0 VMBADDR LSB	0x9f	8	

表C. 11. 2

トップレベルレジスタA. アドレス発生器キーホール (10/19)

【2087】

【表292】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP1 VMBADDR MSB	0xa1	2	テストのみ
BU WADDR COMP1 VMBADDR MID	0xa2	8	
BU WADDR COMP1 VMBADDR LSB	0xa3	8	
BU WADDR COMP2 VMBADDR MSB	0xa5	2	テストのみ
BU WADDR COMP2 VMBADDR MID	0xa6	8	
BU WADDR COMP2 VMBADDR LSB	0xa7	8	
BU WADDR VBADDR MSB	0xa9	2	テストのみ
BU WADDR VBADDR MID	0xaa	8	
BU WADDR VBADDR LSB	0xab	8	
BU WADDR COMP0 HALF WIDTH IN BLOCKS MSB	0xad	2	ロードしな ければなら ない
BU WADDR COMP0 HALF WIDTH IN BLOCKS MID	0xae	8	
BU WADDR COMP0 HALF WIDTH IN BLOCKS LSB	0xaf	8	

表C.11.2

トップレベルレジスタA、アドレス発生器キーホール (11/19)

【2088】

【表293】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP1 HALF WIDTH IN BLOCKS MSB	0xb1	2	ロードしな ければなら ない
BU WADDR COMP1 HALF WIDTH IN BLOCKS MID	0xb2	8	
BU WADDR COMP1 HALF WIDTH IN BLOCKS LSB	0xb3	8	
BU WADDR COMP2 HALF WIDTH IN BLOCKS MSB	0xb5	2	ロードしな ければなら ない
BU WADDR COMP2 HALF WIDTH IN BLOCKS MID	0xb6	8	
BU WADDR COMP2 HALF WIDTH IN BLOCKS LSB	0xb7	8	
BU WADDR HB MSB	0xb8	2	テストのみ
BU WADDR HB MID	0xba	8	
BU WADDR HB LSB	0xbb	8	

表C. 11. 2

トップレベルレジスタA、アドレス発生器キーホール (12/19)

[2089]

【表294】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP0 OFFSET MSB	0xbd	2	ロードしな ければなら ない
BU WADDR COMP0 OFFSET MID	0xbe	8	
BU WADDR COMP0 OFFSET LSB	0xbf	8	
BU WADDR COMP1 OFFSET MSB	0xc1	2	ロードしな ければなら ない
BU WADDR COMP1 OFFSET MID	0xc2	8	
BU WADDR COMP1 OFFSET LSB	0xc3	8	
BU WADDR COMP2 OFFSET MSB	0xc5	2	ロードしな ければなら ない
BU WADDR COMP2 OFFSET MID	0xc6	8	
BU WADDR COMP2 OFFSET LSB	0xc7	8	
BU WADDR SCRATCH MSB	0xc9	2	テストのみ
BU WADDR SCRATCH MID	0xca	8	
BU WADDR SCRATCH LSB	0xcb	8	

表C. 11. 2

トップレベルレジスタA、アドレス発生器キーホール (13/18)

[2090]

【表295】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR MSB WIDE MSB	0xcd	2	ロードしな ければなら ない
BU WADDR MSB WIDE MID	0xce	8	
BU WADDR MSB WIDE LSB	0xcf	8	
BU WADDR MSB HIGH MSB	0xd1	2	ロードしな ければなら ない
BU WADDR MSB HIGH MID	0xd2	8	
BU WADDR MSB HIGH LSB	0xd3	8	
BU WADDR COMP0 LAST MB IN ROW MSB	0xd5	2	ロードしな ければなら ない
BU WADDR COMP0 LAST MB IN ROW MID	0xd6	8	
BU WADDR COMP0 LAST MB IN ROW LSB	0xd7	8	
BU WADDR COMP1 LAST MB IN ROW MSB	0xd9	2	ロードしな ければなら ない
BU WADDR COMP1 LAST MB IN ROW MID	0xda	8	
BU WADDR COMP1 LAST MB IN ROW LSB	0xdb	8	

表C. 11. 2

トップレベルレジスタA、アドレス発生器キーホール (14/19)

[2091]

[表296]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP2 LAST MB IN ROW MSB	0xdd	2	ロードしな ければなら ない
BU WADDR COMP2 LAST MB IN ROW MID	0xde	8	
BU WADDR COMP2 LAST MB IN ROW LSB	0xdf	8	
BU WADDR COMP0 LAST MB IN HALF ROW MSB	0xe1	2	ロードしな ければなら ない
BU WADDR COMP0 LAST MB IN HALF ROW MID	0xe2	8	
BU WADDR COMP0 LAST MB IN HALF ROW LSB	0xe3	8	
BU WADDR COMP1 LAST MB IN HALF ROW MSB	0xe5	2	ロードしな ければなら ない
BU WADDR COMP1 LAST MB IN HALF ROW MID	0xe6	8	
BU WADDR COMP1 LAST MB IN HALF ROW LSB	0xe7	8	

表C. 11. 2

トップレベルレジスタA、アドレス発生器キーホール (15/19)

[2092]

50 [表297]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP2 LAST MB IN HALF ROW MSB	0xe9	2	ロードしな ければなら ない
BU WADDR COMP2 LAST MB IN HALF ROW MID	0xea	8	
BU WADDR COMP2 LAST MB IN HALF ROW LSB	0xeb	8	
BU WADDR COMP0 LAST ROW IN MB MSB	0xed	2	ロードしな ければなら ない
BU WADDR COMP0 LAST ROW IN MB MID	0xee	8	
BU WADDR COMP0 LAST ROW IN MB LSB	0xef	8	
BU WADDR COMP1 LAST ROW IN MB MSB	0xf1	2	ロードしな ければなら ない
BU WADDR COMP1 LAST ROW IN MB MID	0xf2	8	
BU WADDR COMP1 LAST ROW IN MB LSB	0xf3	8	
BU WADDR COMP2 LAST ROW IN MB MSB	0xf5	2	ロードしな ければなら ない
BU WADDR COMP2 LAST ROW IN MB MID	0xf6	8	
BU WADDR COMP2 LAST ROW IN MB LSB	0xf7	8	

表C. 11. 2

トップレベルレジスタA、アドレス発生器キーホール (16/19)

【2093】

【表298】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP0 BLOCKS PER MB ROW MSB	0xf9	2	ロードしな ければなら ない
BU WADDR COMP0 BLOCKS PER MB ROW MID	0xfa	8	
BU WADDR COMP0 BLOCKS PER MB ROW LSB	0xfb	8	
BU WADDR COMP1 BLOCKS PER MB ROW MSB	0xfd	2	ロードしな ければなら ない
BU WADDR COMP1 BLOCKS PER MB ROW MID	0xfe	8	
BU WADDR COMP1 BLOCKS PER MB ROW LSB	0xff	8	
BU WADDR COMP2 BLOCKS PER MB ROW MSB	0x101	2	ロードしな ければなら ない
BU WADDR COMP2 BLOCKS PER MB ROW MID	0x102	8	
BU WADDR COMP2 BLOCKS PER MB ROW LSB	0x103	8	

表C. 11. 2

トップレベルレジスタA、アドレス発生器キーホール (17/19)

[2094]

[表299]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP0 LAST MB ROW MSB	0x105	2	ロードしな ければなら ない
BU WADDR COMP0 LAST MB ROW MID	0x106	8	
BU WADDR COMP0 LAST MB ROW LSB	0x107	8	
BU WADDR COMP1 LAST MB ROW MSB	0x109	2	
BU WADDR COMP1 LAST MB ROW MID	0x10a	8	ロードしな ければなら ない
BU WADDR COMP1 LAST MB ROW LSB	0x10b	8	
BU WADDR COMP2 LAST MB ROW MSB	0x10d	2	
BU WADDR COMP2 LAST MB ROW MID	0x10e	8	
BU WADDR COMP2 LAST MB ROW LSB	0x10f	8	ロードしな ければなら ない
BU WADDR COMP0 HBS MSB	0x111	2	
BU WADDR COMP0 HBS MID	0x112	8	
BU WADDR COMP0 HBS LSB	0x113	8	

表C. 11. 2

トップレベルレジスタA. アドレス発生器キーホール (18/19)

[2095]

[表300]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU WADDR COMP1 HBS MSB	0x115	2	ロードしな ければなら ない
BU WADDR COMP1 HBS MID	0x116	8	
BU WADDR COMP1 HBS LSB	0x117	8	
BU WADDR COMP2 HBS MSB	0x119	2	ロードしな ければなら ない
BU WADDR COMP2 HBS MID	0x11a	8	
BU WADDR COMP2 HBS LSB	0x11b	8	
BU WADDR COMP0 MAXHB	0x11f	2	
BU WADDR COMP1 MAXHB	0x123	2	ロードしな ければなら ない
BU WADDR COMP2 MAXHB	0x127	2	
BU WADDR COMP0 MAXVB	0x12b	2	
BU WADDR COMP1 MAXVB	0x12f	2	ロードしな ければなら ない
BU WADDR COMP2 MAXVB	0x133	2	

表C. 11. 2

トップレベルレジスタA. アドレス発生器キーホール (19/19)

[2096]

50 [表301]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU UH0 A00 1	0x0	5	R/W-係数0, 0
BU UH0 A00 0	0x1	8	
BU UH0 A01 1	0x2	5	R/W-係数0, 1
BU UH0 A01 0	0x3	8	
BU UH0 A02 1	0x4	5	R/W-係数0, 2
BU UH0 A02 0	0x5	8	
BU UH0 A03 1	0x6	5	R/W-係数0, 0
BU UH0 A03 0	0x7	8	
BU UH0 A10 1	0x8	5	R/W-係数1, 0
BU UH0 A10 0	0x9	8	
BU UH0 A11 1	0xa	5	R/W-係数1, 1
BU UH0 A11 0	0xb	8	
BU UH0 A12 1	0xc	5	R/W-係数1, 2
BU UH0 A12 0	0xd	8	
BU UH0 A13 1	0xe	5	R/W-係数1, 3
BU UH0 A13 0	0xf	8	

表C. 11. 3

H-アップサンプラ及びCspaceキーホールアドレスマップ (1/7)

[2097]

[表302]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU UH0 A20 1	0x10	5	R/W-係数2, 0
BU UH0 A20 0	0x11	8	
BU UH0 A21 1	0x12	5	R/W-係数2, 1
BU UH0 A21 0	0x13	8	
BU UH0 A22 1	0x14	5	R/W-係数2, 2
BU UH0 A22 0	0x15	8	
BU UH0 A23 1	0x16	5	R/W-係数2, 3
BU UH0 A23 0	0x17	8	
BU UH0 MODE	0x18	2	R/W
BU UH0 A00 1	0x20	5	R/W-係数0, 0
BU UH0 A00 0	0x21	8	
BU UH1 A01 1	0x22	5	R/W-係数0, 1
BU UH1 A01 0	0x23	8	
BU UH1 A02 1	0x24	5	R/W-係数0, 2
BU UH1 A02 0	0x25	8	

表C. 11. 3

H-アップサンプラ及びCspaceキーホールアドレスマップ (2/7)

[2098]

[表303]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU UH1 A03 1	0x26	5	R/W-係数0, 0
BU UH1 A03 0	0x27	8	
BU UH1 A10 1	0x28	5	R/W-係数1, 0
BU UH1 A10 0	0x29	8	
BU UH1 A11 1	0x2a	5	R/W-係数1, 1
BU UH1 A11 0	0x2b	8	
BU UH1 A12 1	0x2c	5	R/W-係数1, 2
BU UH1 A12 0	0x2d	8	
BU UH1 A13 1	0x2e	5	R/W-係数1, 3
BU UH1 A13 0	0x2f	8	
BU UH1 A20 1	0x30	5	R/W-係数2, 0
BU UH1 A20 0	0x31	8	
BU UH1 A21 1	0x32	5	R/W-係数2, 1
BU UH1 A21 0	0x33	8	
BU UH1 A22 1	0x34	5	R/W-係数2, 2
BU UH1 A22 0	0x35	8	

表C. 11. 3

H-アップサンブラ及びCspaceキーホールアドレスマップ (3/7)

[2099]

【表304】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU UH1 A23 1	0x36	5	R/W-係数2, 3
BU UH1 A23 0	0x37	8	
BU UH1 MODE	0x38	2	R/W
BU UH2 A00 1	0x40	5	R/W-係数0, 0
BU UH2 A00 0	0x41	8	
BU UH2 A01 1	0x42	5	R/W-係数0, 1
BU UH2 A01 0	0x43	8	
BU UH2 A02 1	0x44	5	R/W-係数0, 2
BU UH2 A02 0	0x45	8	
BU UH2 A03 1	0x46	5	R/W-係数0, 0
BU UH2 A03 0	0x47	8	
BU UH2 A10 1	0x48	5	R/W-係数1, 0
BU UH2 A10 0	0x49	8	
BU UH2 A11 1	0x4a	5	R/W-係数1, 1
BU UH2 A11 0	0x4b	8	

表C. 11. 3

H-アップサンブラ及びCspaceキーホールアドレスマップ (4/7)

[2100]

【表305】

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU UH2 A12 1	0x4c	5	R/W-係数 1, 2
BU UH2 A12 0	0x4d	8	
BU UH2 A13 1	0x4e	5	R/W-係数 1, 3
BU UH2 A13 0	0x4f	8	
BU UH2 A20 1	0x50	5	R/W-係数 2, 0
BU UH2 A20 0	0x51	8	
BU UH2 A21 1	0x52	5	R/W-係数 1
BU UH2 A21 0	0x53	8	
BU UH2 A22 1	0x54	5	R/W-係数 2, 2
BU UH2 A22 0	0x55	8	
BU UH2 A23 1	0x56	5	R/W-係数 2, 3
BU UH2 A23 0	0x57	8	
BU UH2 MODE	0x58	2	R/W
BU CS A00 1	0x60	5	R/W
BU CS A00 0	0x61	8	

表C. 11. 3

H-アップサンブラ及びCspaceキーホールアドレスマップ (5/7)

[2101]

[表306]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU CS A10 1	0x62	5	R/W
BU CS A10 0	0x63	8	
BU CS A20 1	0x64	5	R/W
BU CS A20 0	0x65	8	
BU CS B0 1	0x66	5	R/W
BU CS B0 0	0x67	8	
BU CS A01 1	0x68	5	R/W
BU CS A01 0	0x69	8	
BU CS A11 1	0x6a	5	R/W
BU CS A11 0	0x6b	8	
BU CS A21 1	0x6c	5	R/W
BU CS A21 0	0x6d	8	
BU CS B1 1	0x6e	5	R/W
BU CS B1 0	0x6f	8	
BU CS A02 1	0x70	5	R/W
BU CS A02 0	0x71	8	

表C. 11. 3

H-アップサンブラ及びCspaceキーホールアドレスマップ (6/7)

[2102]

[表307]

キーホールレジスタ名	キーホール アドレス	ビット	コメント
BU CS A12 1	0x72	5	R/W
BU CS A12 0	0x73	8	
BU CS A22 1	0x74	5	R/W
BU CS A22 0	0x75	8	
BU CS B2 1	0x76	5	R/W
BU CS B2 0	0x77	8	

表C. 11. 3

H-アップサンブラ及びCspaceキーホールアドレスマップ (7/7)

667

668

以下の様式化されたコード片は、ライトアドレス発生器からのピクチャサイズ割り込みに答えるために必要なプロセッシングを示す。ピクチャサイズパラメータは、HORIZONTAL MBSトークン、VERTICAL MBSトークン、及び（各コンポーネントのために）DEFINE SAMPLINGトークンを送り、その結果としてライトアドレス発生器割り込みを生じさせることにより、on-the-flyで変更できることに注意。これらのトークンは如何なるオーダーで到着してもよく、一般的に、いずれもがピクチャサイズパラメータの全ての再計算を必要とするべきである。しかしながら、セットアップ時に、計算を実行する前に、全てのイベントの到着を検出する方が効率的であろう。

【2103】セットアップ時に、ピクチャサイズパラメータレジスタに特別なバリューを書き込むことができるので、トークンに答える割り込み処理に依存する必要がある。このため、SIFピクチャ用の適当なレジスタバリューも与えられる。

【2104】C. 13. 2 ピクチャサイズパラメータ用の割り込み処理
5個のピクチャサイズイベントがあり、各々の応答の主たるものを下記に記す：

```

if (hmbs event)
    load (mbs wide);
else if (vmbs event)
    load (mbs high);
else if (def samp0 event)
{
    load (maxhb [0]);
    load (maxvb [0]);
}
else if (def samp1 event)
{
    load (maxhb [1]);
    load (maxvb [1]);
}
else if (def samp2 event)
{
    load (maxhb [2]);
    load (maxvb [2]);
}

```

20 それに加えて、一貫したピクチャサイズパラメータを保持するためには、以下の計算が必要である：

```

if (hmbs event || vmbs event ||
    def samp0 event || def samp1 event ||
    def samp2 event)
{
    for (i=0; i<max component; i++)
    {
        hbs [i] = addr hbs [i] =
            (maxhb [i] + 1) mbs wide;
        half width in blocks [i] =
            ((maxhb [i] + 1) mbs wide) / 2;
        last mb in row [i] =
            hbs [i] - (maxhb [i] + 1);
        last mb in half row [i] =
            half width in blocks [i]
            - (maxhb [i] + 1);
        last row in mb [i] =
            hbs [i] maxvb [i];
        blocks per mb row [i] =
            last row in mb [i] + hbs [i];
        last mb row [i] =
            blocks per mb row [i]
            (mbs high - 1);
    }
}

```

ピクチャサイズパラメータに答えて、（表示ウインドーサイズ等の）dispaddrレジスタバリューを修正することが厳密に必要であるが、これはアプリケーションの必要条件に依存することが望ましい。

【2105】C. 13. 3 「SIFピクチャ用のレジスタバリュー」

SIF用の上記割り込み処理の後、全てのピクチャサイズレジスタに含まれるバリュー、4:2:0ストリームは以下の通りである：

C. 13. 3. 1 「一次バリュー」
BU WADDR MBS WIDE=0x16
BU WADDR MBS HIGH=0x12

669

670

BU WADDR COMP0 MAXHB=0x01
 BU WADDR COMP1 MAXHB=0x00
 BU WADDR COMP2 MAXHB=0x00
 BU WADDR COMP0 MAXVB=0x01
 BU WADDR COMP1 MAXVB=0x00
 BU WADDR COMP2 MAXVB=0x00
 C. 13. 3. 2 「二次バリュウー計算後」
 BU WADDR COMP0 HBS=0x2C
 BU WADDR COMP1 HBS=0x16
 BU WADDR COMP2 HBS=0x16
 BU ADDR COMP0 HBS=0x2C
 BU ADDR COMP1 HBS=0x16
 BU ADDR COMP2 HBS=0x16
 BU WADDR COMP0 HALF WIDTH
 IN BLOCKS=0x16
 BU WADDR COMP1 HALF WIDTH
 IN BLOCKS=0x03
 BU WADDR COMP2 HALF WIDTH
 IN BLOCKS=0x03
 BU WADDR COMP0 LAST MB IN
 ROW=0x2A
 BU WADDR COMP1 LAST MB IN
 ROW=0x15
 BU WADDR COMP2 LAST MB IN
 ROW=0x15
 BU WADDR COMP0 LAST MB IN
 HALF ROW=0x14
 BU WADDR COMP1 LAST MB IN
 HALF ROW=0x0A
 BU WADDR COMP2 LAST MB IN
 HALF ROW=0x0A
 BU WADDR COMP0 LAST ROW I
 N MB=0x2C
 BU WADDR COMP1 LAST ROW I
 N MB=0x0
 BU WADDR COMP2 LAST ROW I
 N MB=0x0
 BU WADDR COMP0 BLOCKS PER
 MB ROW=0x58
 BU WADDR COMP1 BLOCKS PER
 MB ROW=0x16
 BU WADDR COMP2 BLOCKS PER
 MB ROW=0x16
 BU WADDR COMP0 LAST MB RO
 W=0x5D8
 BU WADDR COMP1 LAST MB RO
 W=0x176
 BU WADDR COMP2 LAST MB RO
 W=0x176

これらのバリュウーがセットアップ時にはつきりと書き込

まれることになっている場合、ほとんどのロケーションの多重バイト特性を考慮に入れなければならないことに注意。

【2106】上述してきた本発明のパイプラインシステムは、長い間望まれてきたプロセッシングステージにおける制御及び/もしくはデータ機能のための、複数のプロセッシングステージ及び対話式インターフェイスングトークンの形態での万能順応装置を持つ改良システムであり、それによってプロセッシングステージは構成及びプロセッシングのフレキシビリティを高められる。処理ステージは少なくとも1つのトークンにตอบสนองして構成可能である。処理ステージの1つは入力を受信してトークンを発生/変換するスタートコード検出器である。

【2107】改良パイプラインシステムはビデオデータ用であり、ハフマンデコードと、インデックス/データユニット、演算論理ユニットと、複数の異なるピクチャ圧縮/伸長規格の各々に対する記憶された個別プログラムを持つマイクロコードROMとを具備し、プログラムはトークンによって選択可能である空間デコードを具備し、複数の異なるピクチャ圧縮/伸長規格に対する処理を容易とする。

【2108】本発明の多重規格ビデオ伸長装置は、パイプラインプロセッシングマシンとして配置される2線式インターフェースにより相互接続される複数のステージを持ち、制御トークン及びデータトークンはトークンフォーマットで制御とデータの両方を運ぶために1つの2線式インターフェースを通過する。トークンデコード回路は、そのステージに関する制御トークンとして特定のトークンを認識し、パイプラインに沿って未認識の制御トークンを送るために、特定のステージに配置される。再配置プロセッシング回路は選ばれたステージに配置され、特定されたデータトークンを処理するために該かるステージを再配置するための認識済み制御トークンに反応する。該かるシステムを実装するために、広範囲に亘る独特なサブ支持システム回路とプロセッシング技術を開示する。

【2109】上述したことから、発明の特別な形態を図示して説明してきたが、発明の精神及び範囲を逸脱することなく、様々な修正が可能であることが自明であろう。従って、添付クレームによる制限以外は、図示した形態によって本発明が制限されるものではない。

【2110】

【発明の効果】以上説明したように、本発明によれば、複数の処理ステージと、処理ステージの中で制御、データ機能を実行する相互作用インターフェース制御トークンの形態のユニバーサル適応ユニットを具備し、処理ステージが構成、処理において向上した柔軟性を与えられているパイプラインシステムが提供される。

【図面の簡単な説明】

【図1】2個の内部制御信号の異なる組み合わせのため

の 6 段パイプラインの 6 サイクルを示す図。

【図 2】パイプラインステージがパイプライン内の遅延に
応答して圧縮、及び伸長が出来る方法を示すために各
ステージが二次データ格納部を含むパイプラインを示す
図。

【図 3】パイプラインステージがパイプライン内の遅延
に
応答して圧縮、及び伸長が出来る方法を示すために各
ステージが二次データ格納部を含むパイプラインを示す
図。

【図 4】2 配線インターフェースと多相クロックを使用
10 するパイプラインの好ましい実施例のステージ間のデー
タ転送の制御を示す図。

【図 5】2 配線インターフェースと多相クロックを使用
するパイプラインの好ましい実施例のステージ間のデー
タ転送の制御を示す図。

【図 6】2 配線インターフェースと多相クロックを使用
するパイプラインの好ましい実施例のステージ間のデー
タ転送の制御を示す図。

【図 7】2 配線インターフェースと多相クロックを使用
20 するパイプラインの好ましい実施例のステージ間のデー
タ転送の制御を示す図。

【図 8】2 配線インターフェースが組み込まれたパイプ
ラインステージの基本的実施例及び 2 配線転送制御を有
する 2 個の連続するパイプライン処理ステージを示すブ
ロック図。

【図 9】図 8 に示すパイプラインステージで使用される
タイミング信号と、入出力データと、内部制御信号間の
関係を示すタイミング図の一例。

【図 10】図 8 に示すパイプラインステージで使用され
るタイミング信号と、入出力データと、内部制御信号間
30 の関係を示すタイミング図の一例。

【図 11】拡張ビットの制御下でその状態を維持するパ
イプラインステージの一例を示すブロック図。

【図 12】ステージ起動データワードを復号化するパイ
プラインステージのブロック図。

【図 13】データ二重化パイプラインステージの一例で
の 2 配線転送制御の使用を示すブロック図。

【図 14】データ二重化パイプラインステージの一例で
の 2 配線転送制御の使用を示すブロック図。

【図 15】図 13 と図 14 に例示する実施例で使用され
40 る 2 相クロックと、2 配線転送制御信号と、その他の内
部データ及び制御信号を示すタイミング図の一例。

【図 16】図 13 と図 14 に例示する実施例で使用され
る 2 相クロックと、2 配線転送制御信号と、その他の内
部データ及び制御信号を示すタイミング図の一例。

【図 17】再構成可能な処理ステージのブロック図。

【図 18】空間デコーダのブロック図。

【図 19】時間デコーダのブロック図。

【図 20】ビデオフォーマッタのブロック図。

【図 21】マクロブロックの第 1 の構成を示すメモリマ
50 ップ。

ップ。

【図 22】マクロブロックの第 2 の構成を示すメモリマ
ップ。

【図 23】マクロブロックの別の構成を示すメモリマ
ップ。

【図 24】ありうるテーブル選択値のベン図。

【図 25】本発明で使用する画像データの変長を示す
図。

【図 26】予測フィルタを含む一時的デコーダのブロッ
ク図。

【図 27】予測フィルタ処理を絵で示した図。

【図 28】マクロブロックの構造を示す一般的な図。

【図 29】開始コード検出器の一般的なブロック図。

【図 30】データストリーム内の開始コードを例示する
図。

【図 31】フラグ発生器と、復号化インデックスと、ヘ
ッド発生器と、追加ワード発生器と、出力ラッチの關係
を示すブロック図。

【図 32】空間デコーダ DRAM インターフェースのブ
ロック図。

【図 33】書き込みスイングバッファのブロック図。

【図 34】処理中のブロックからの予測データオフセッ
トを示す挿絵図。

【図 35】(1、1) の予測データオフセットを示す
図。

【図 36】ハフマンデコーダと空間デコーダの分割状態
装置を示すブロック図。

【図 37】予測フィルタのブロック図。

【図 38】典型的なデコーダシステムを示す図。

【図 39】J P E G 静止画像デコーダを示す図。

【図 40】J P E G ビデオデコーダを示す図。

【図 41】多重規格ビデオデコーダを示す図。

【図 42】トークンの開始と終了を示す図。

【図 43】トークンアドレスとデータフィールドを示す
図。

【図 44】8 ビットより広いインターフェース上のトー
クンを示す図。

【図 45】マクロブロックを示す図。

【図 46】2 配線インターフェースプロトコルを示す
図。

【図 47】外部 2 配線インターフェースの位置を示す
図。

【図 48】クロック伝搬を示す図。

【図 49】2 配線インターフェースのタイミングを示す
図。

【図 50】アクセス構造を例示する。

【図 51】読出し転送サイクルを示す図。

【図 52】アクセス開始タイミングを示す図。

【図 53】2 回の書き込み転送のアクセス例を示す図。

【図 54】読出し転送サイクルを示す図。

【図55】書込み転送サイクルを示す図。

【図56】リフレッシュサイクルを示す図。

【図57】32ビットデータバスと256kビットの深さを持つDRAM(9ビットアドレス)の例を示す図。

【図58】ストローブ信号用のタイミングパラメータを示す図。

【図59】どの2個のストローブ信号間のタイミングパラメータを示す図。

【図60】バスとストローブ信号間のタイミングパラメータを示す図。

【図61】バスとストローブ信号間のタイミングパラメータを示す図。

【図62】MPI読出しタイミングを示す図。

【図63】MPI書込みタイミングを示す図。

【図64】メモリマップの大きな整数の配置を示す図。

【図65】典型的なデコードクロック形態を示す図。

【図66】入力クロック要件を示す図。

【図67】空間デコードを示す図。

【図68】入力回路の入出力を示す図。

【図69】符号化ポートプロトコルを示す図。

【図70】開始コード検出器を示す図。

【図71】検出された開始コードと変換されたトークンを示す図。

【図72】開始コード検出器がトークンを通すところを示す図。

【図73】MPEG開始コード(バイト整列)の重なりを示す図。

【図74】MPEG開始コード(非バイト整列)の重なりを示す図。

【図75】2個のビデオ系列間のジャンプを示す図。

【図76】追加のトークンの挿入処理を示す図。

【図77】デコード起動制御を示す図。

【図78】出力前にキューされたイネーブルされたストリームを示す図。

【図79】空間デコードバッファを示す図。

【図80】バッファポインタを示す図。

【図81】ビデオデマルチプレクサを示す図。

【図82】画像構造を示す図。

【図83】4:2:2マクロブロックの構造を示す図。

【図84】マクロブロックの大きさ画素の大きさから計算する方法を示す図。

【図85】空間デコードを示す図。

【図86】H.261逆量子化の概観を示す図。

【図87】JPEG逆量子化の概観を示す図。

【図88】MPEG逆量子化の概観を示す図。

【図89】量子化テーブルメモリマップを示す図。

【図90】JPEGベースライン系列構造の概観を示す図。

【図91】トークンされたJPEG画像を示す図。

【図92】時間デコードを示す図。

【図93】画像バッファ仕様を示す図。

【図94】MPEG画像系列(m=3)を示す図。

【図95】「I」画像がどのようにして格納出力されるかを示す図。

【図96】「P」画像がどのようにして格納出力されるかを示す図。

【図97】「B」画像がどのようにして格納出力されるかを示す図。

【図98】ピクチャバッファを示す図。

10 【図99】H.261予測形式を示す図。

【図100】H.261系列を示す図。

【図101】H.261シンタックスの階層を示す図。

【図102】H.261画像層を示す図。

【図103】ブロック群のH.261配置を示す図。

【図104】H.261「スライス」層を示す図。

【図105】マクロブロックのH.261配置を示す図。

【図106】H.261ブロック系列を示す図。

【図107】H.261マクロブロック層を示す図。

20 【図108】画素のH.261配置を示す図。

【図109】MPEGシンタックスの階層を示す図。

【図110】MPEG系列層を示す図。

【図111】MPEG群の画像層を示す図。

【図112】MPEG画像層を示す図。

【図113】MPEG「スライス」層を示す図。

【図114】MPEG系列のブロックを示す図。

【図115】MPEGマクロブロック層を示す図。

【図116】「オープンGOP」の例を示す図。

【図117】アクセス構造の例を示す図。

30 【図118】アクセス開始タイミングを示す図。

【図119】高速ページ読出しサイクルを示す図。

【図120】高速ページ書込みサイクルを示す図。

【図121】リフレッシュサイクルを示す図。

【図122】チップアドレスからいかにして行及び列アドレスを抽出するかを示す図。

【図123】ストローブ信号用のタイミングパラメータを示す図。

【図124】どの2個のストローブ信号間のタイミングパラメータを示す図。

【図125】バスとストローブ信号間のタイミングパラメータを示す図。

【図126】バスとストローブ信号間のタイミングパラメータを示す図。

【図127】ハフマンデコードとパーザーを示す図。

【図128】H.261とMPEG AC係数復号化フローチャートを示す図。

【図129】JPEG(ACとDC)係数復号化のブロック図。

50 【図130】JPEG(ACとDC)係数復号化の流れ図を示す図。

675

【図131】ハフマントークンフォーマットのインタフェースを示す図。

【図132】トークンフォーマットのブロック図。

【図133】H.261とMPEG AC係数復号化を示す図。

【図134】ハフマンALUのインタフェースを示す図。

【図135】ハフマンALUの基本的構造を示す図。

【図136】バッファマネージャを示す図。

【図137】imodelとhsppkのブロック図。 10

【図138】imex状態を示す図。

【図139】バッファスタートアップを示す図。

【図140】DRAMインターフェースを示す図。

【図141】書き込みスイングバッファを示す図。

【図142】演算ブロックを示す図。

【図143】iqのブロック図。

【図144】iqcアステートマシンを示す図。

【図145】IDCT 1-D変換アルゴリズムを示す図。

【図146】IDCT 1-D変換構造を示す図。 20

【図147】トークンストリームのブロック図。

【図148】標準のブロック図。

【図149】マイクロプロセッサテストアクセスを示すブロック図。

【図150】1-D変換マイクロ構造を示す図。

【図151】時間デコーダのブロック図。

【図152】2配線インターフェースステージの構造を示す図。

【図153】アドレス発生器のブロック図。

【図154】ブロックと画素のオフセットを示す図。 30

【図155】複数の予測フィルタを示す図。

【図156】予測フィルタを示す図。

【図157】1-D予測フィルタを示す図。

676

【図158】1ブロックの画素を示す図。

【図159】読出しラダーの構造を示す図。

【図160】ブロックと画素のオフセットを示す図。

【図161】予測例を示す図。

【図162】読出しサイクルを示す図。

【図163】書き込みサイクルを示す図。

【図164】タイミングを参照可能なトップレベルレジスタ群のブロック図。

【図165】プレゼンテーション番号を増加させる制御を示す図。

【図166】バッファマネージャステートマシンを示す図。

【図167】ステートマシンのメインループを示す図。

【図168】SIF (22×18のマクロブロック) を含むバッファ0を示す図。

【図169】表示窓を持つSIF成分0を示す図。

【図170】格納ブロックアドレスを示す画像フォーマット例を示す図。

【図171】SIF (22×18のマクロブロック) を含むバッファ0を示す図。

【図172】アドレス計算例を示す図。

【図173】書き込みアドレス発生状態を示す図。

【図174】データ路のスライスを示す図。

【図175】データ路の2サイクル操作を示す図。

【図176】モード1のフィルタ処理を示す図。

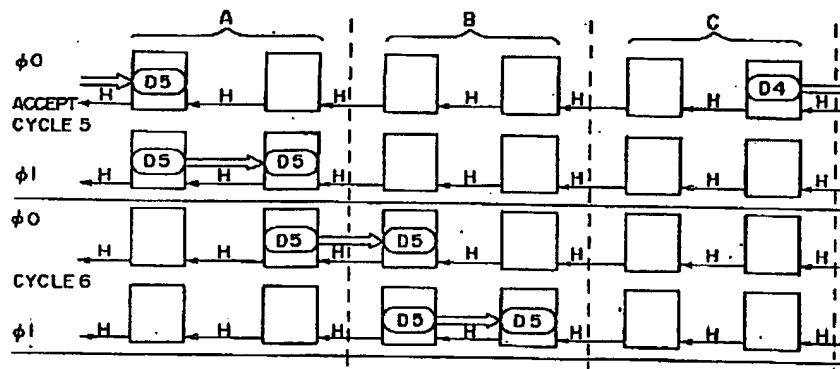
【図177】水平アップサンブラデータ路を示す図。

【図178】色-空間変換器の構造を示す図。

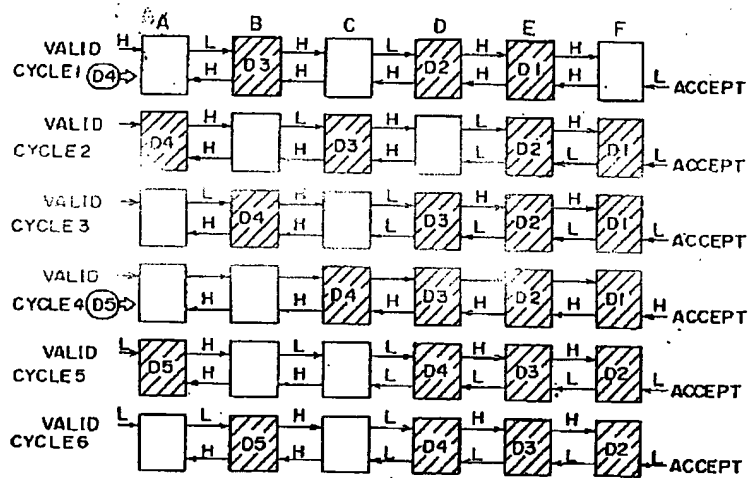
【符号の説明】

33…トークンデコーダ、34…入力ラッチ、36…プロセッシングユニット、39…アクション識別部、41…出力ラッチ、43、44…レジスタ。

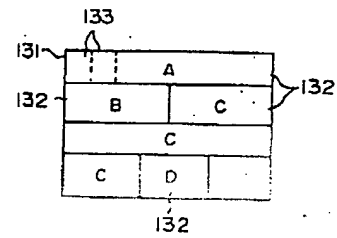
【図6】



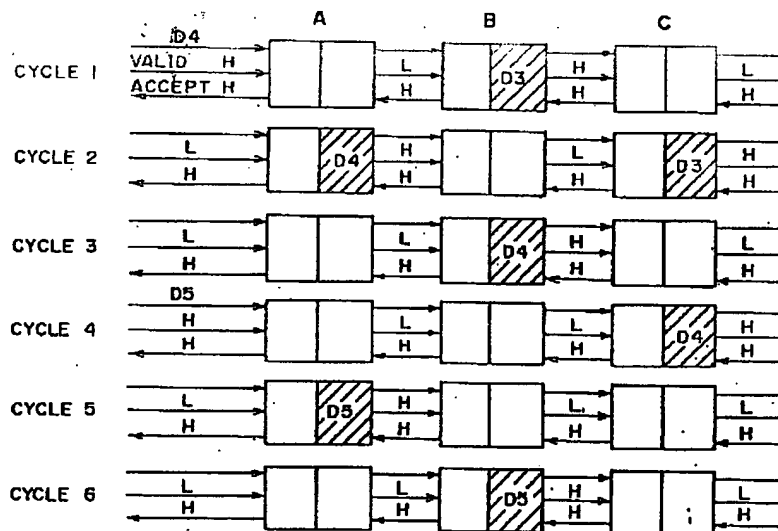
【図 1】



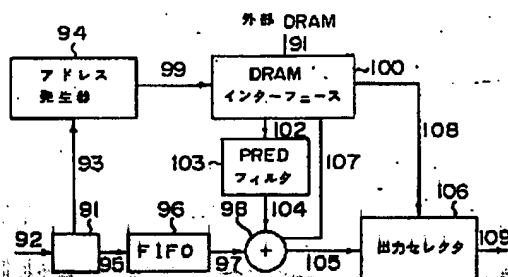
【図 2 1】



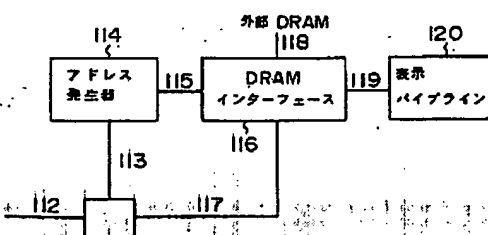
【図2】



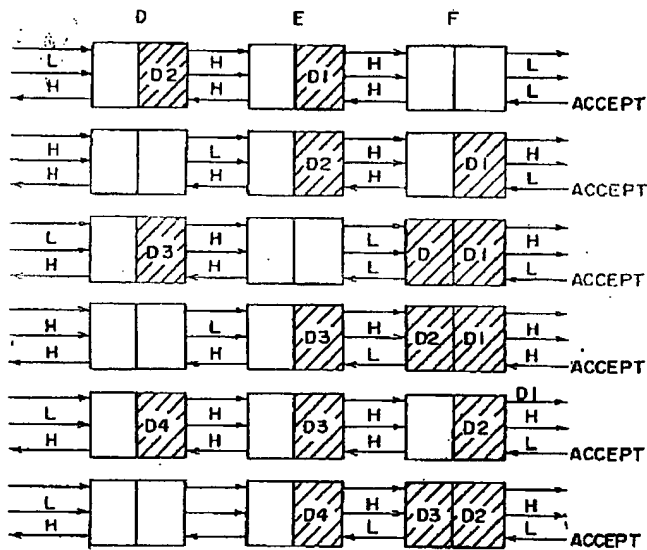
【图 19】



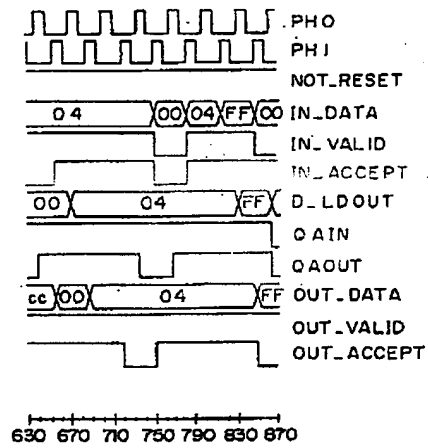
【图20】



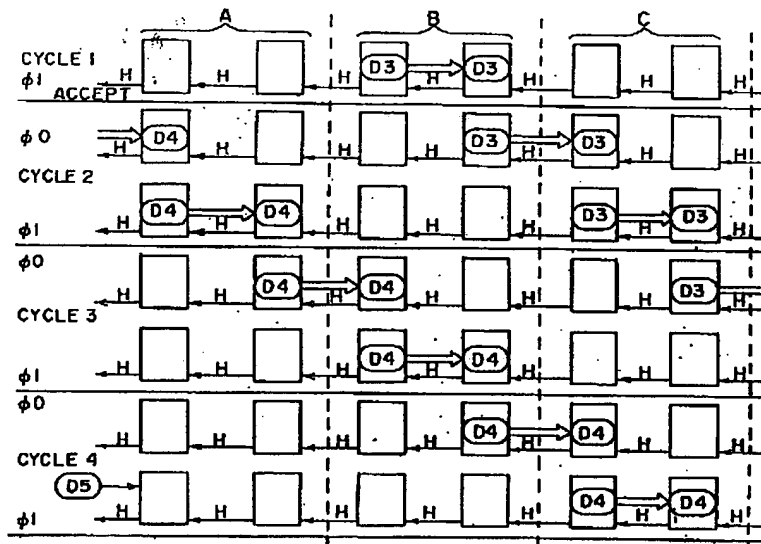
【図 3】



【図 10】

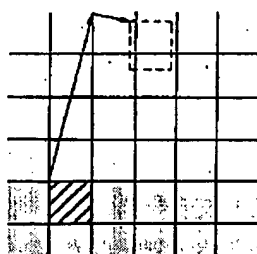
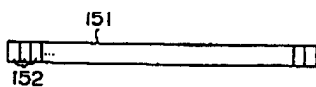


【図 4】

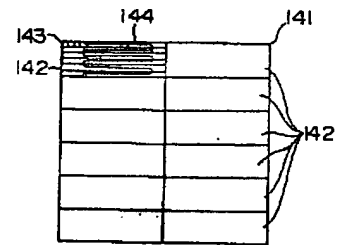


【図 23】

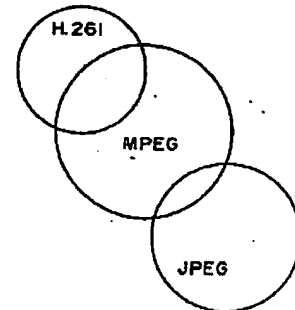
【図 34】



【図 22】



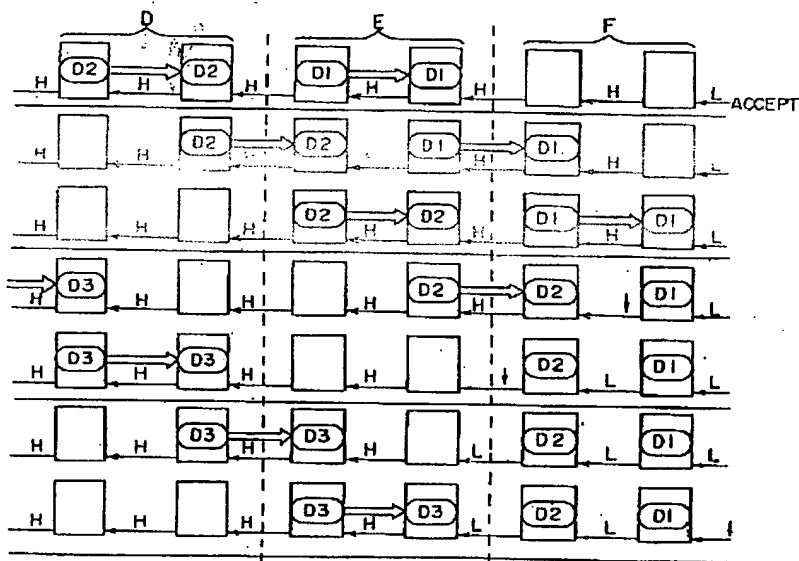
【図 24】



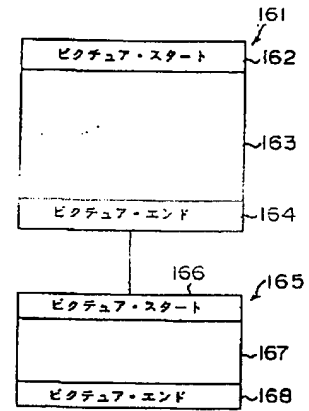
【図 39】



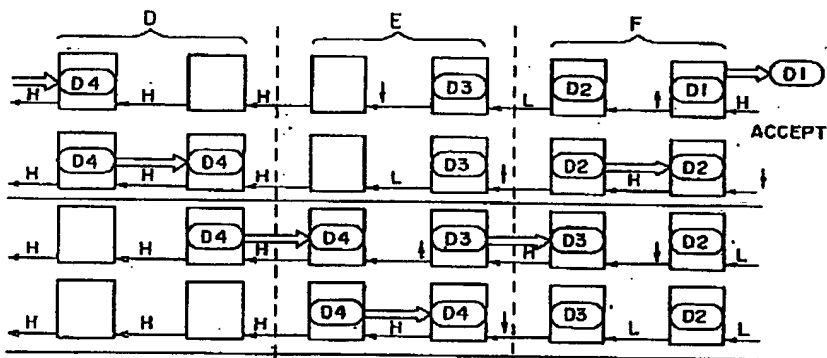
【図 5】



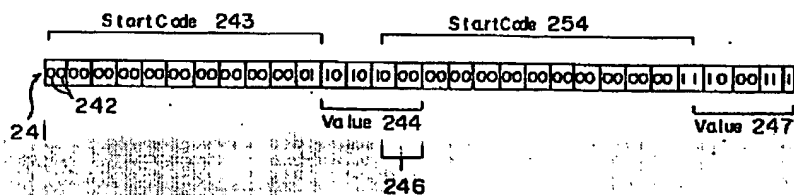
【図 25】



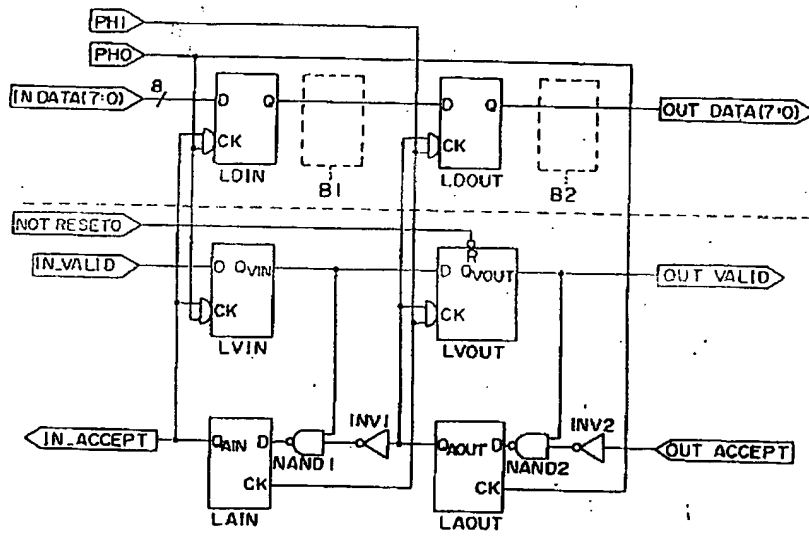
【図 7】



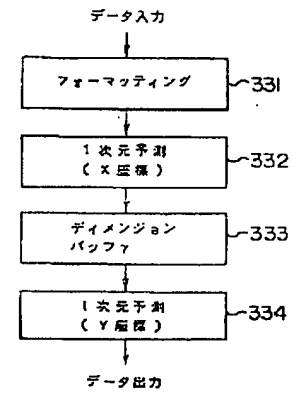
【図 30】



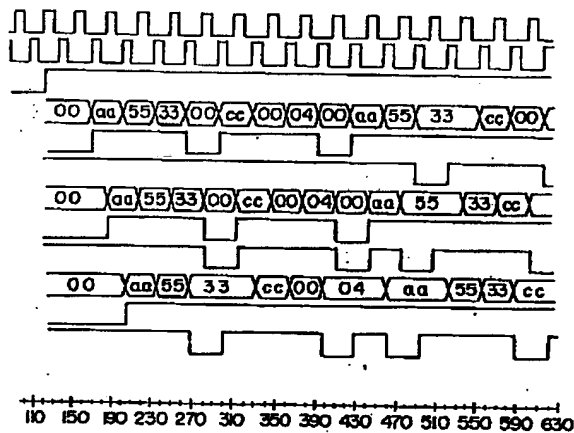
【図 8】



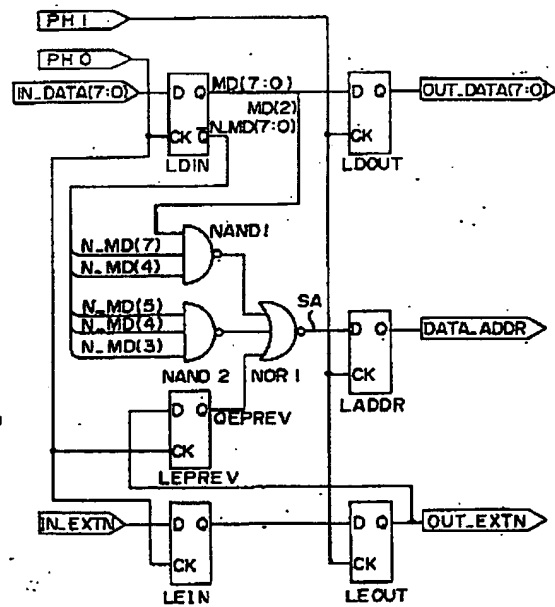
【図 37】



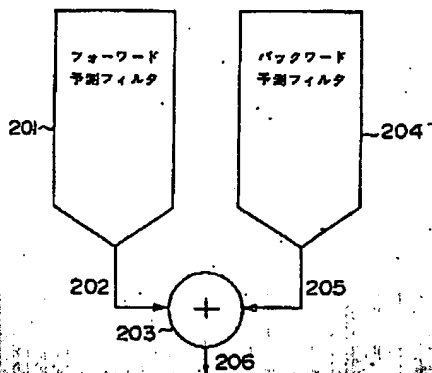
【図 9】



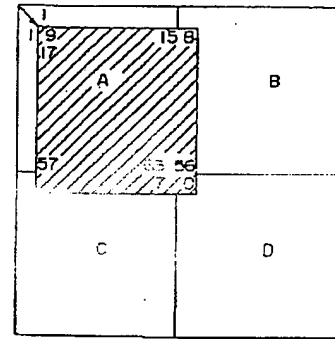
【図 11】



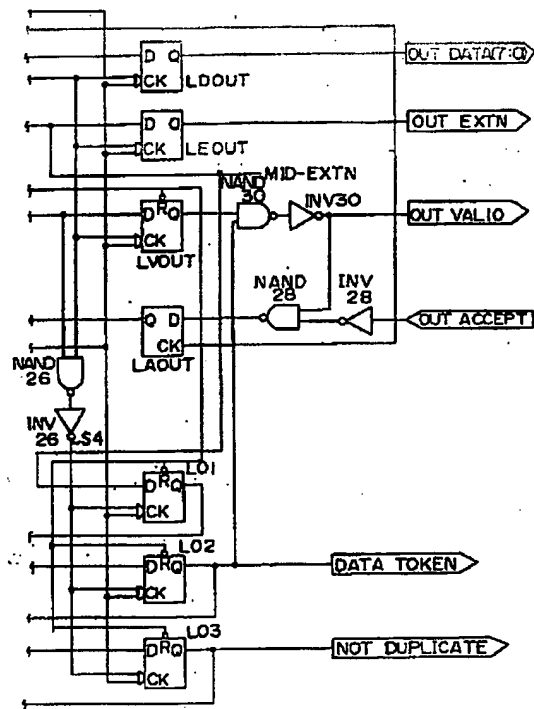
【図 27】



【図 35】



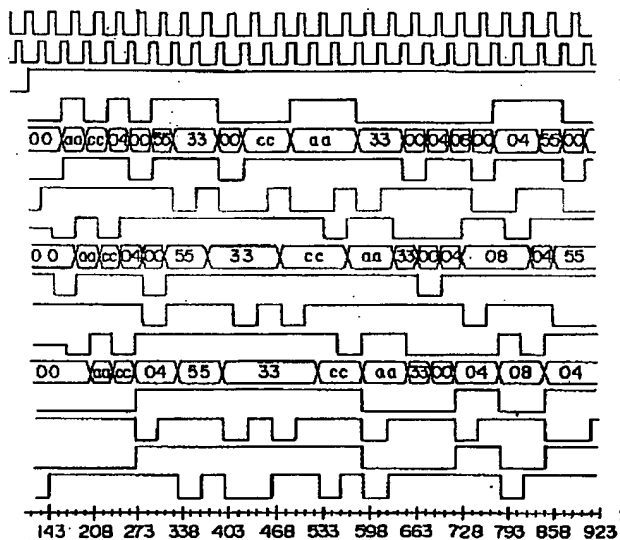
【图 14】



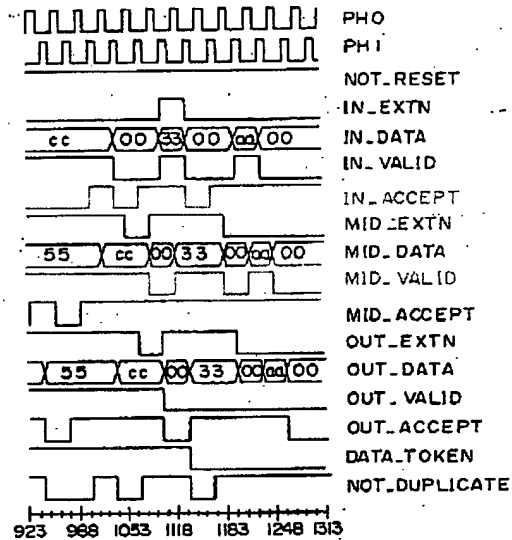
【图 4 1】



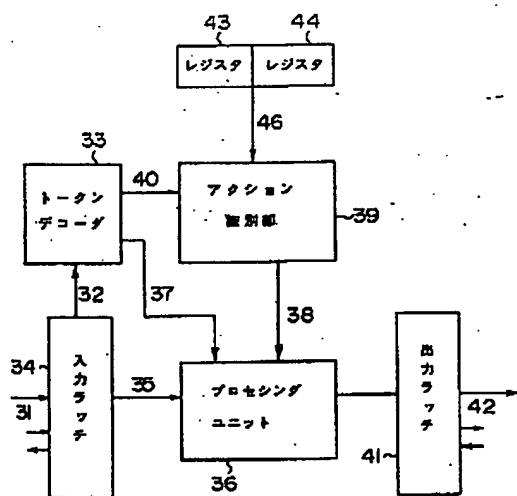
【図 15】



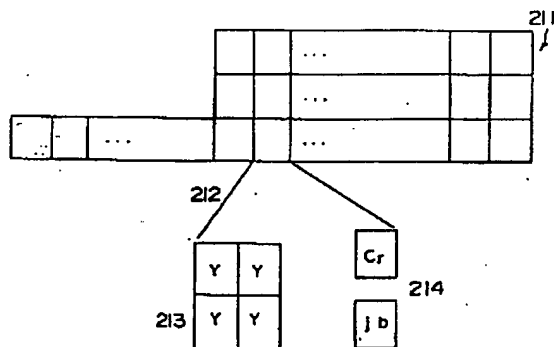
【図 16】



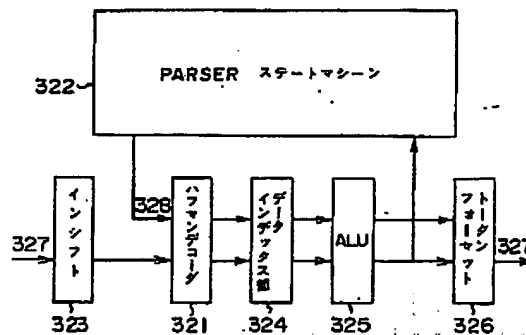
【図 17】



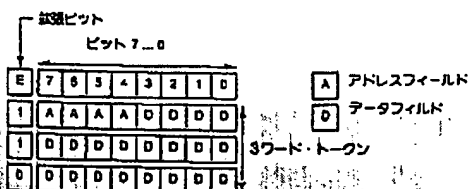
【図 28】



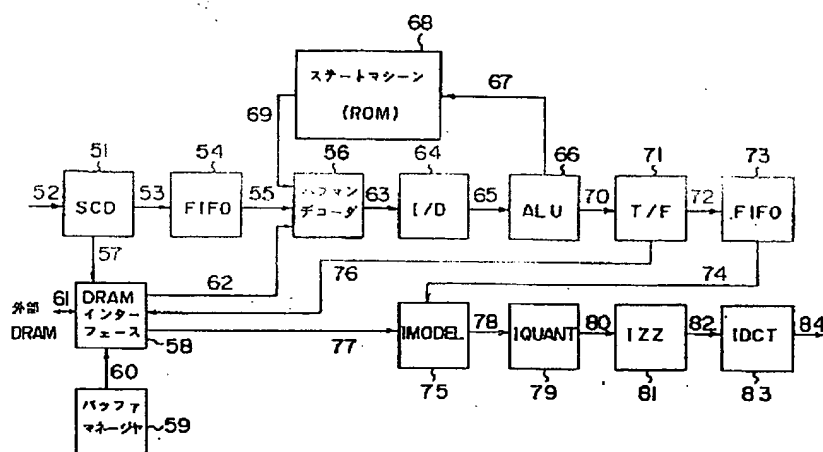
【図 36】



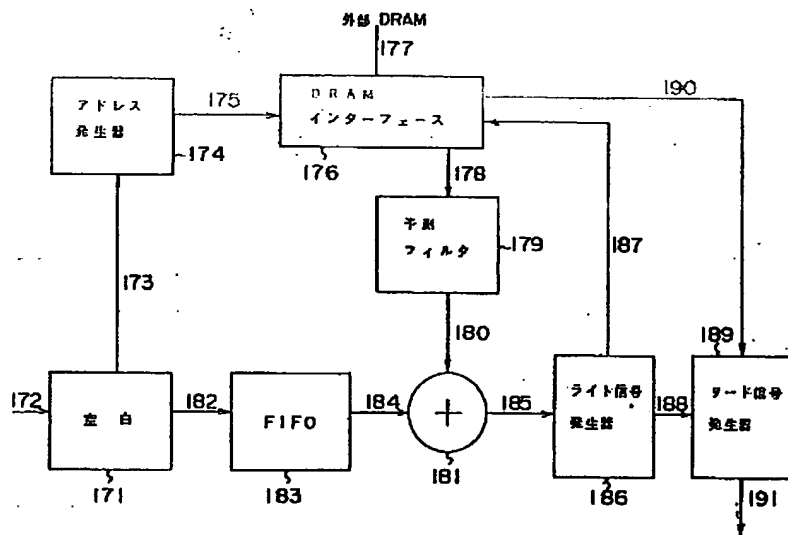
【図 43】



【图 18】

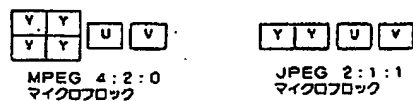
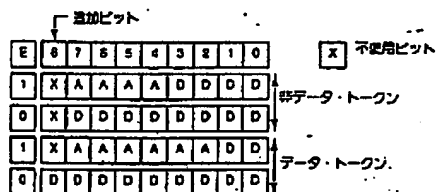


【图 2 6】

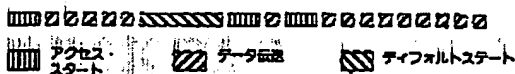


【图 4-4】

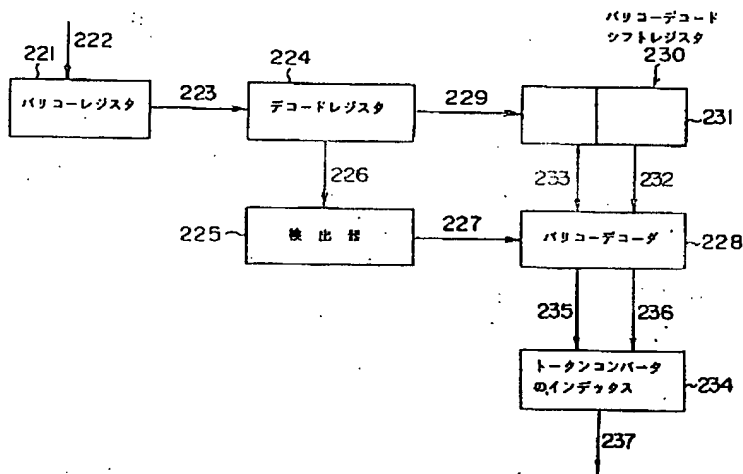
【图 4 5】



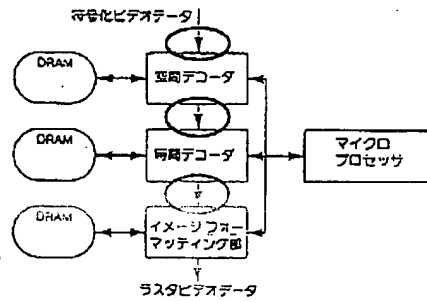
【图 50】



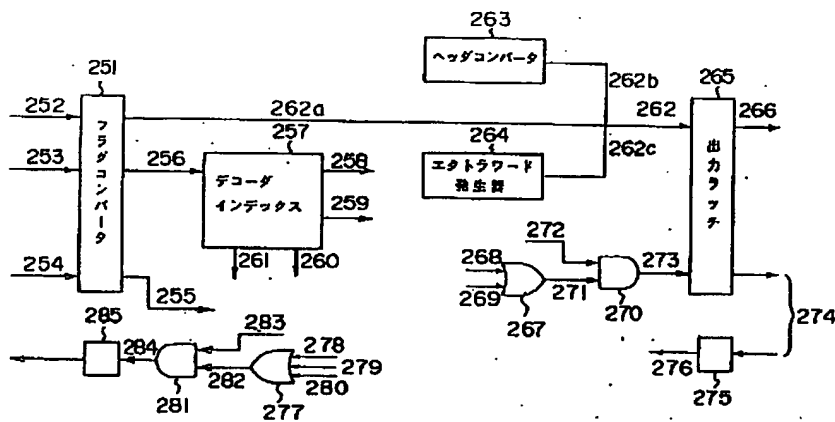
【図 29】



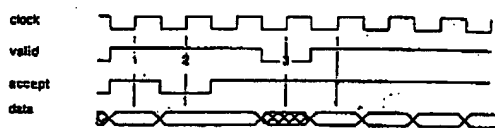
【図 47】



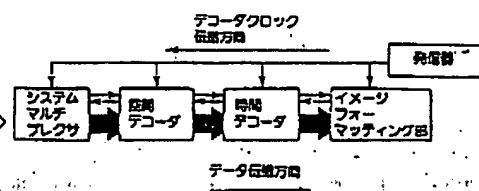
【図 31】



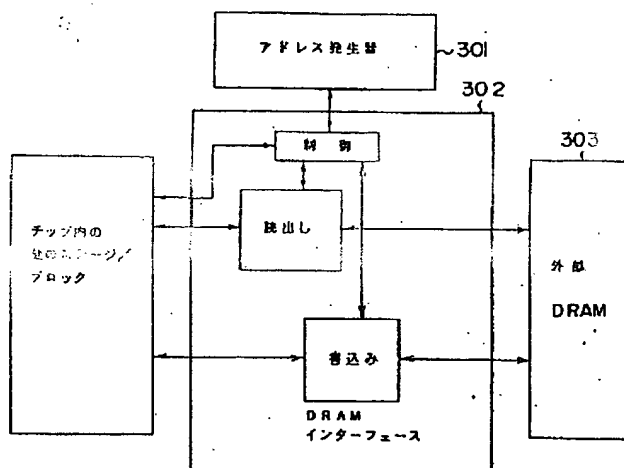
【図 46】



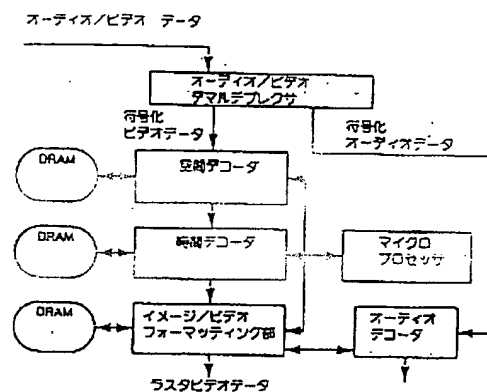
【図 48】



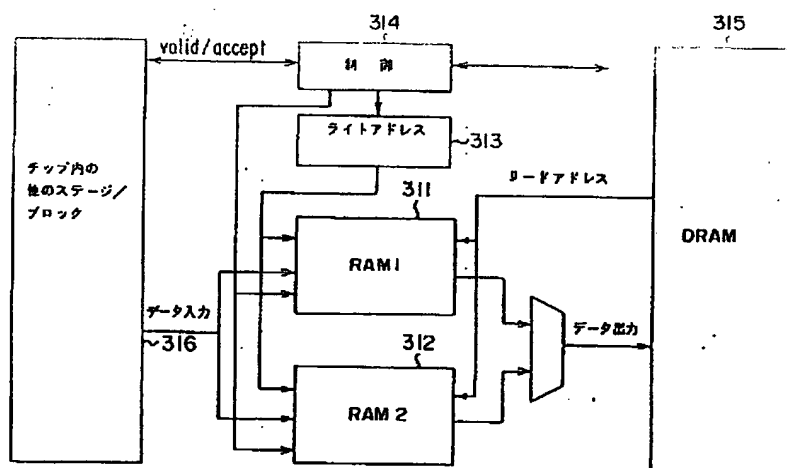
【図 32】



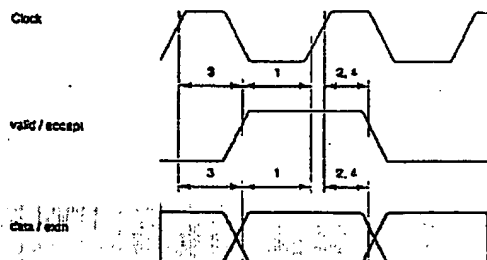
【図 38】



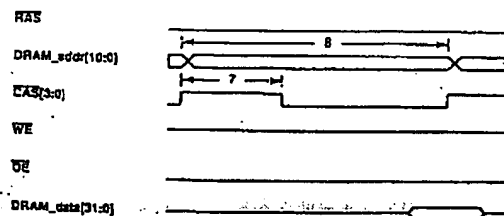
【図 33】



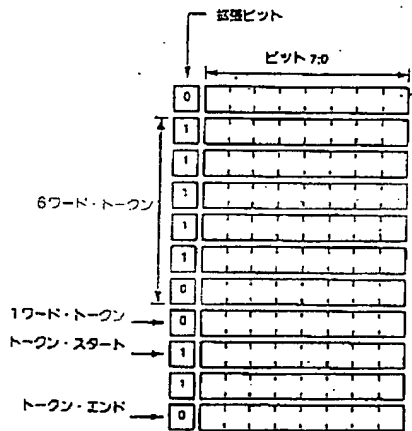
【図 49】



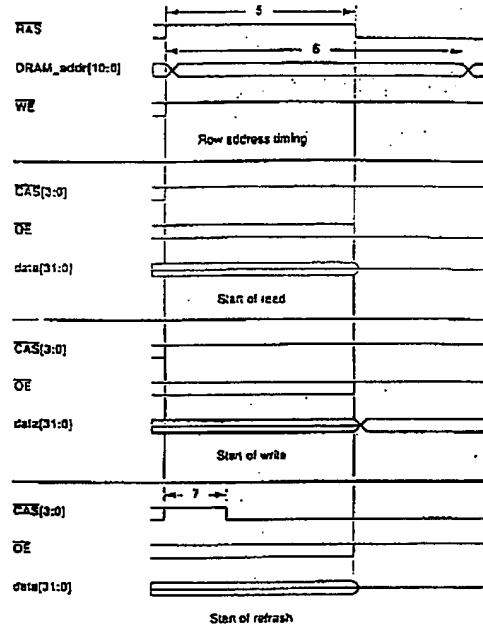
【図 51】



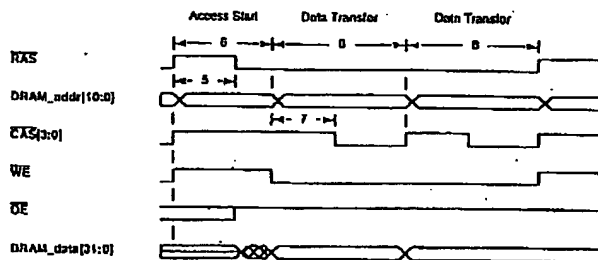
【図42】



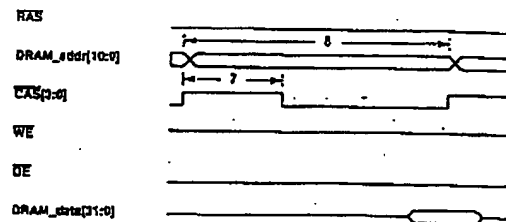
【図52】



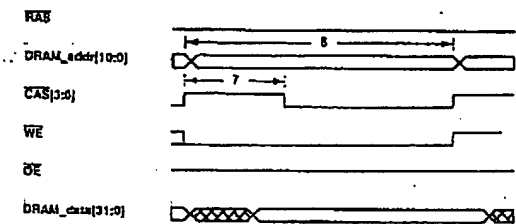
【図53】



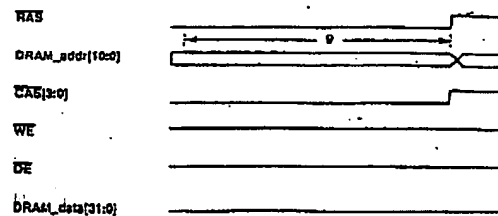
【図54】



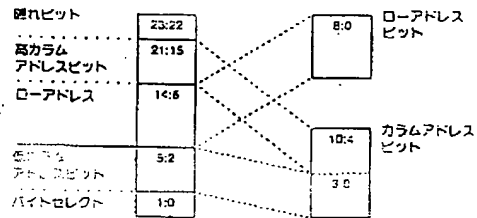
【図55】



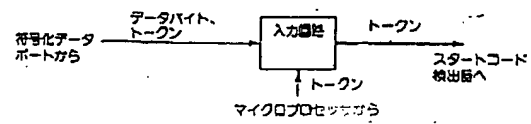
【図56】



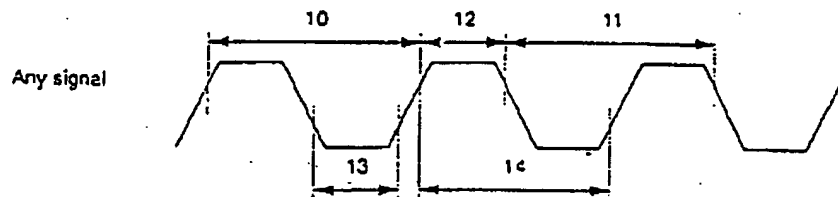
【図57】



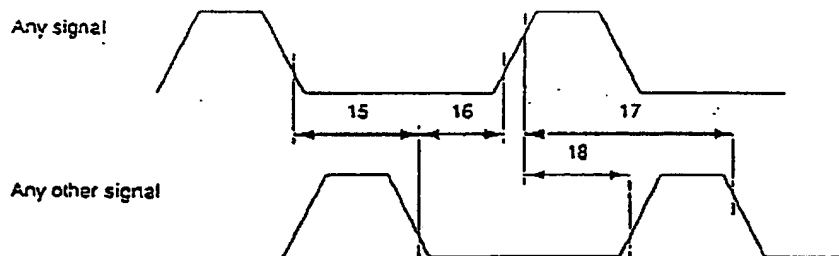
【図68】



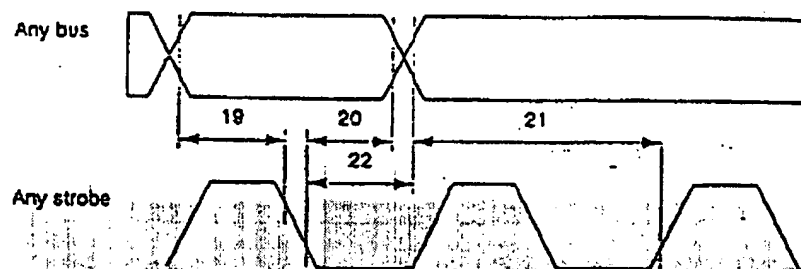
【図58】



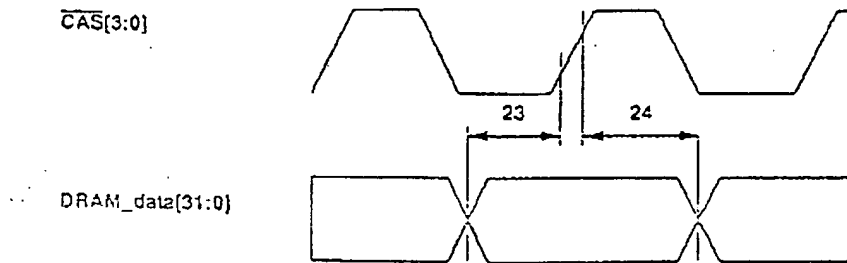
【図59】



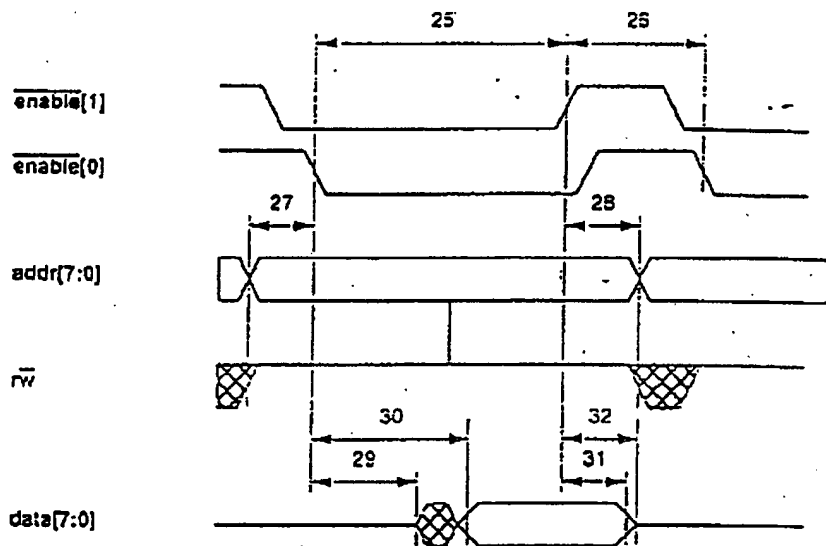
【図60】



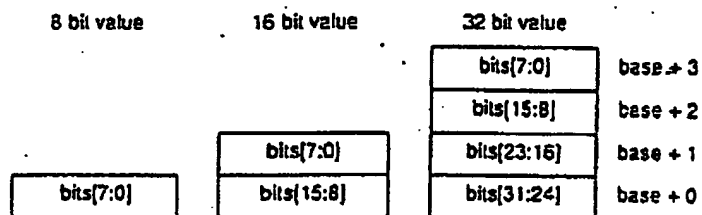
【図 6 1】



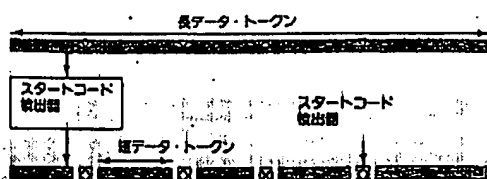
【図 6 2】



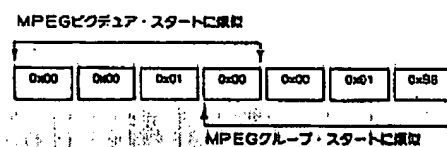
【図 6 4】



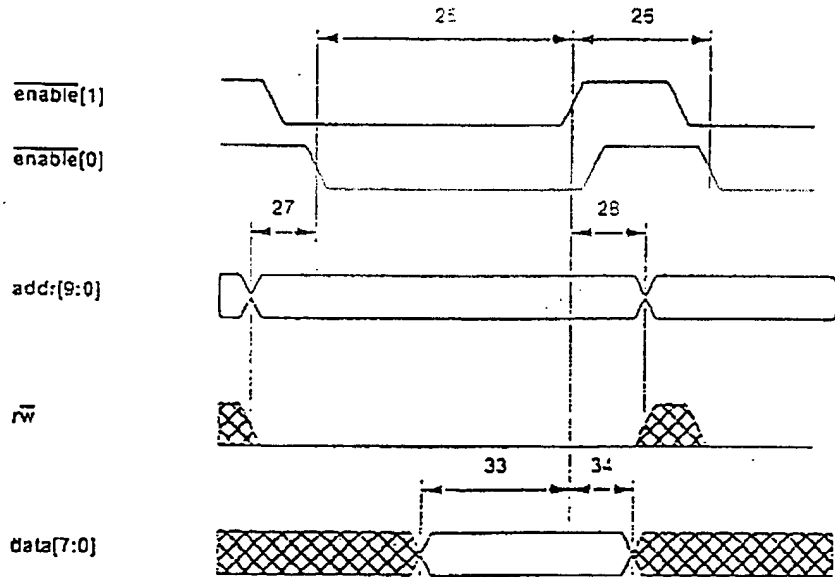
【図 7 1】



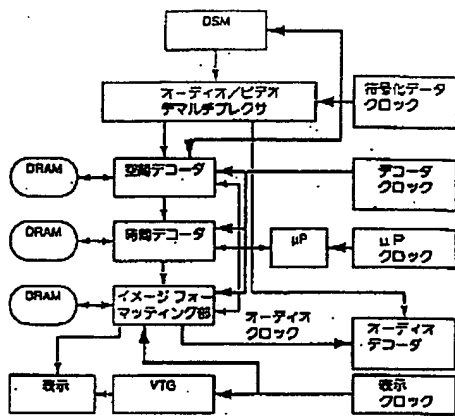
【図 7 3】



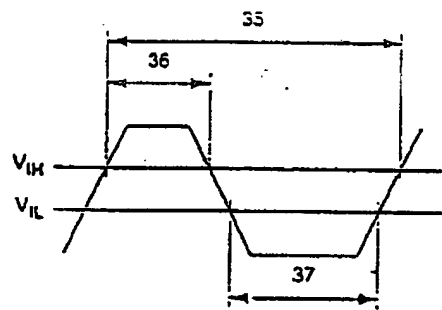
【図63】



【図65】

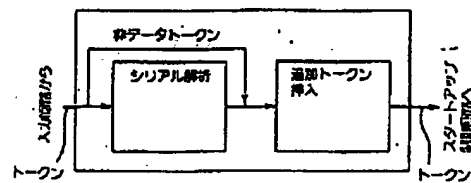
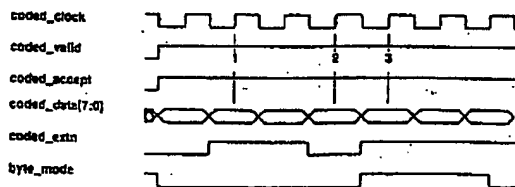


【図66】



【図70】

【図69】

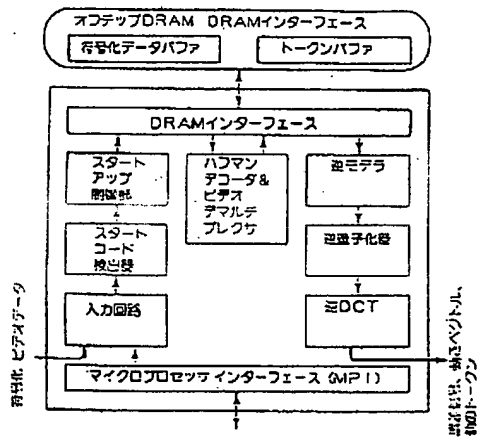


【図84】

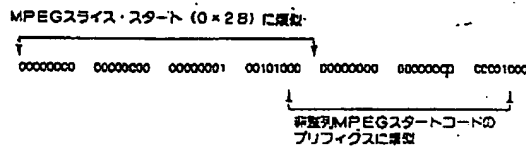
$$\text{水平マクロブロック数} = \frac{\text{水平ピクセル数} + 15}{16}$$

$$\text{垂直マクロブロック数} = \frac{\text{垂直ピクセル数} + 15}{16}$$

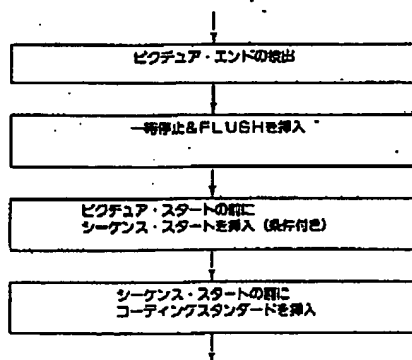
【図67】



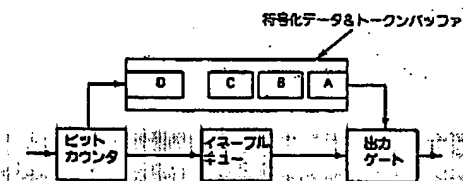
【図74】



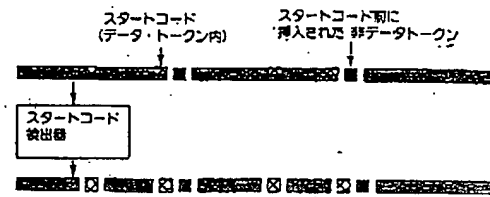
【図76】



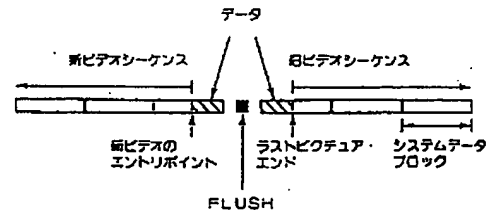
【図78】



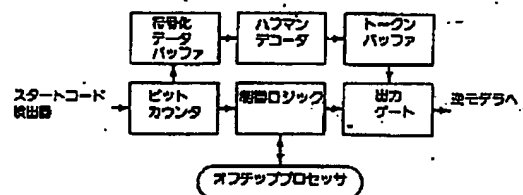
【図72】



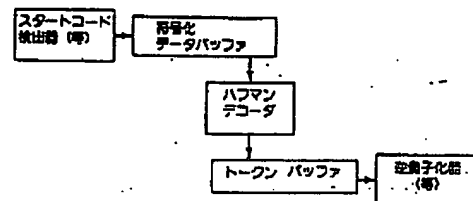
【図75】



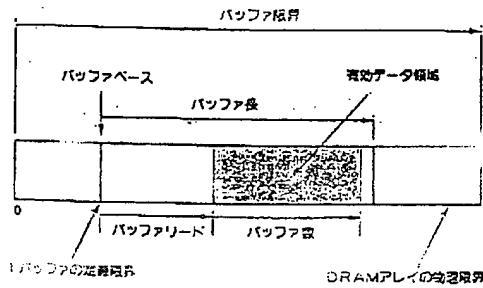
【図77】



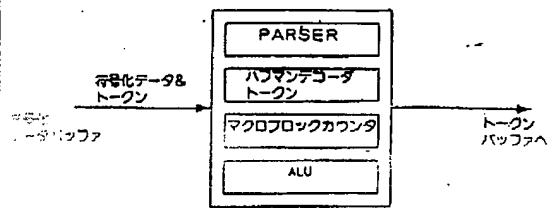
【図79】



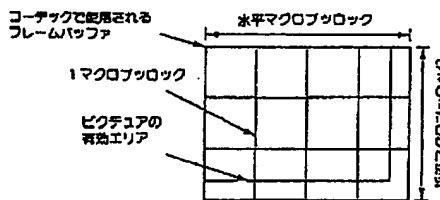
【図80】



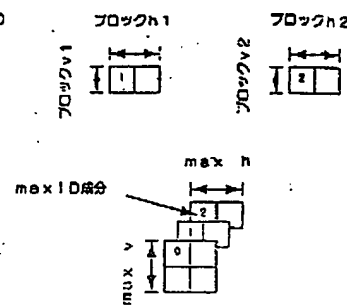
【図81】



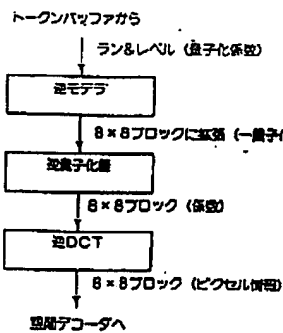
【図82】



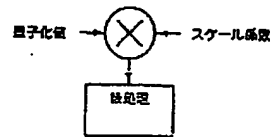
【図83】



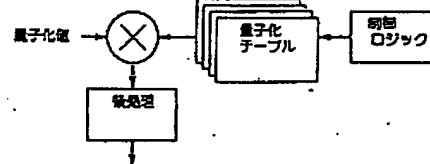
【図85】



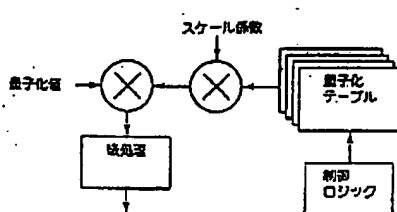
【図86】



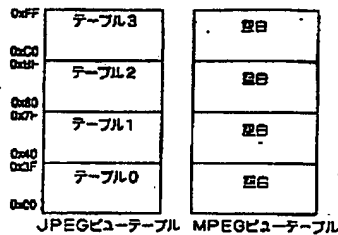
【図87】



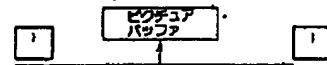
【図88】



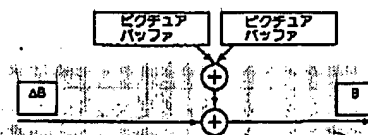
【図89】



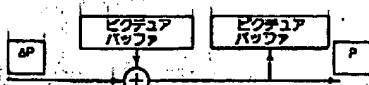
【図95】



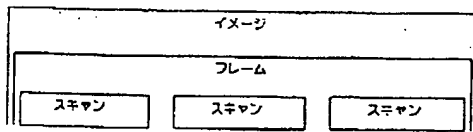
【図97】



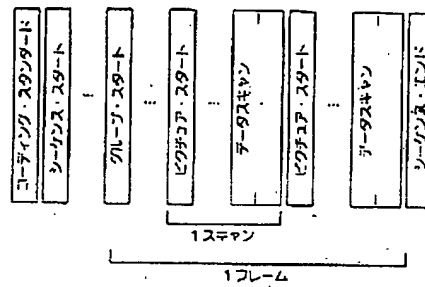
【図96】



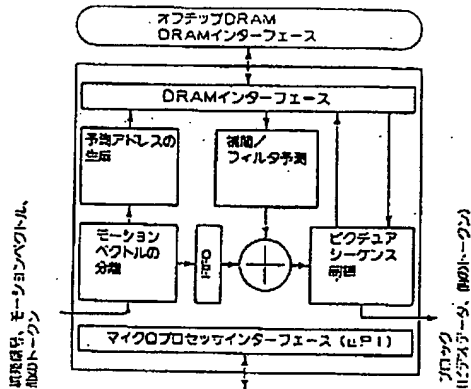
【図 90】



【図 91】

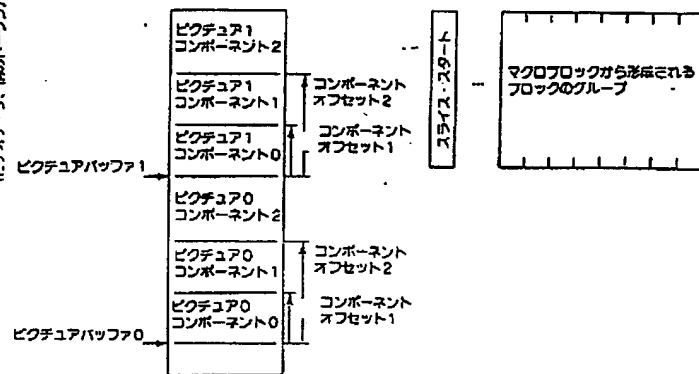


【図 92】



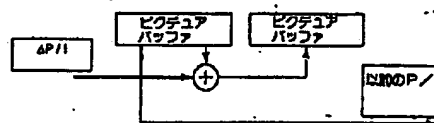
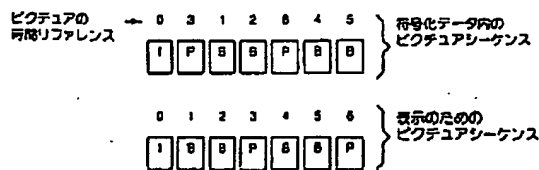
【図 93】

【図 104】



【図 94】

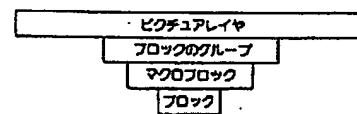
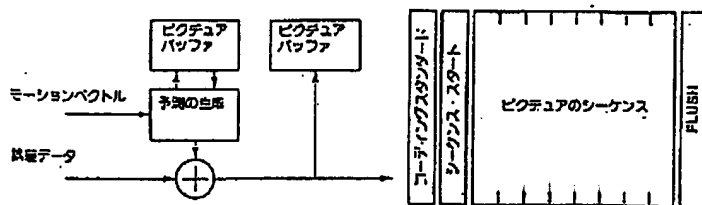
【図 98】



【図 99】

【図 100】

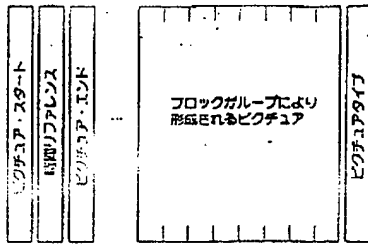
【図 101】



【図 105】

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	30	31	32	33

【図102】



【図103】

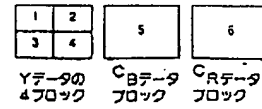
OF

0	1
2	3
4	5
6	7
8	9
10	11

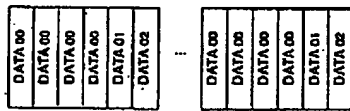
OCF

0
2
4

【図106】



【図107】

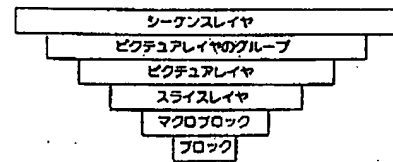


【図108】

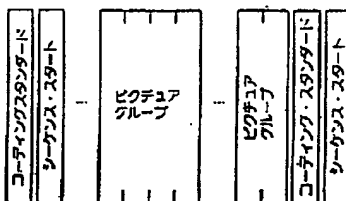
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16

59	58	59	60	61	62	63	64
----	----	----	----	----	----	----	----

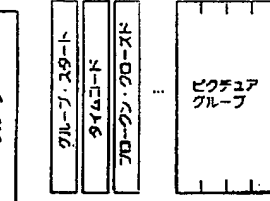
【図109】



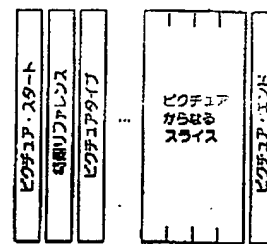
【図110】



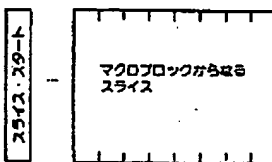
【図111】



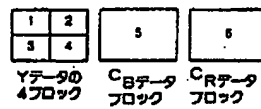
【図112】



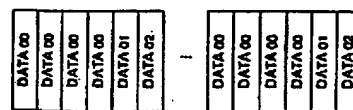
【図113】



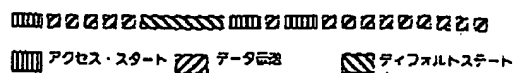
【図114】



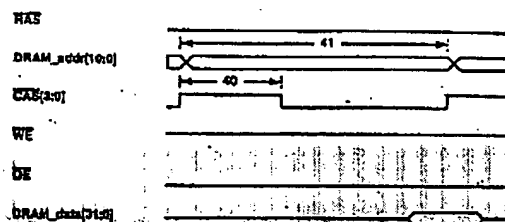
【図115】



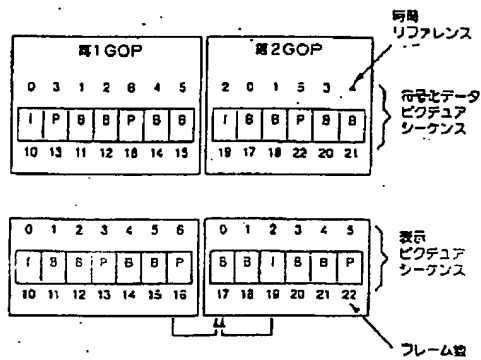
【図117】



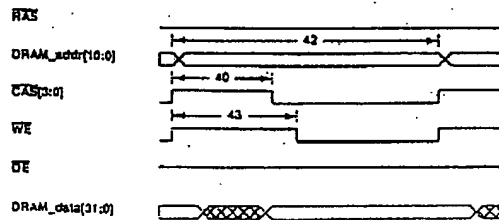
【図119】



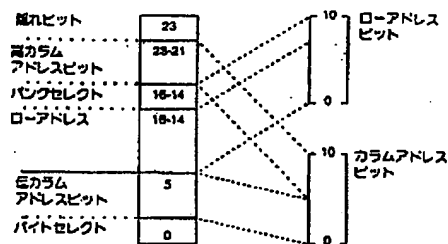
【図116】



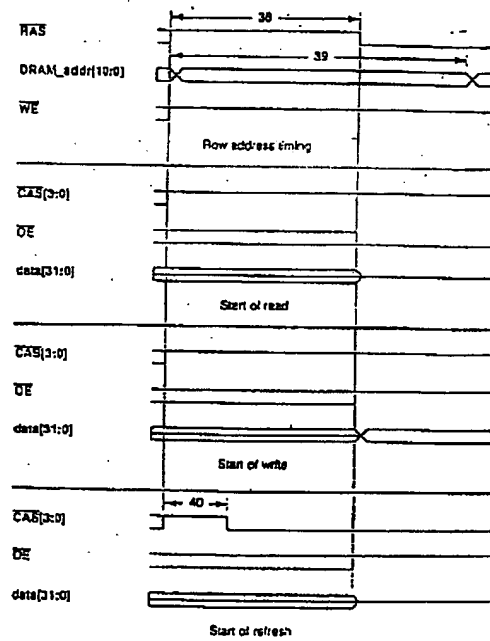
【図120】



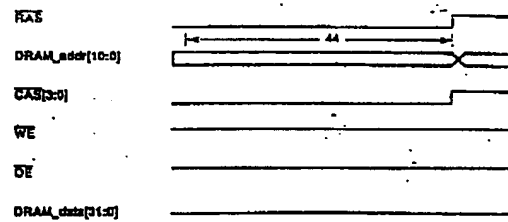
【図122】



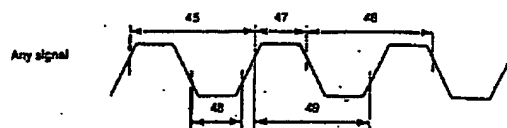
【図118】



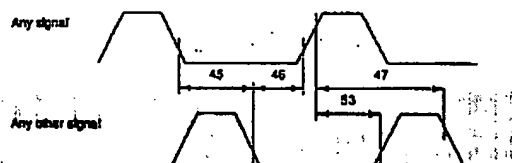
【図121】



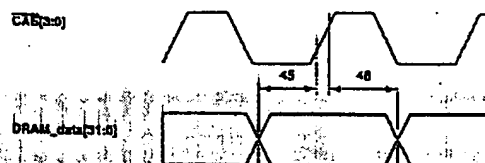
【図123】



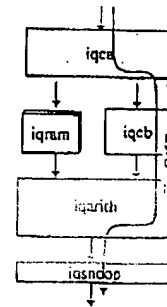
【図124】



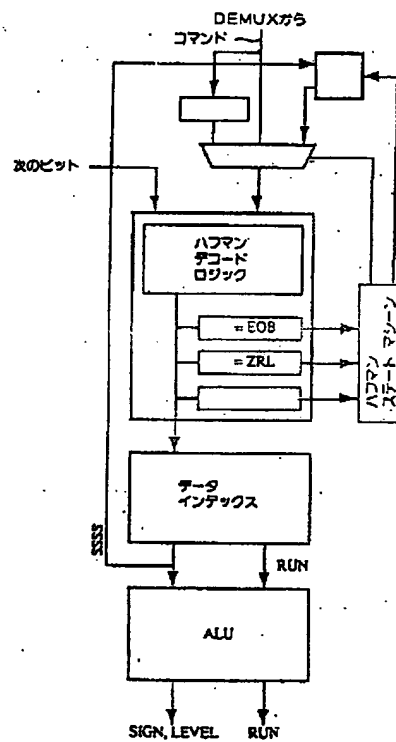
【図126】



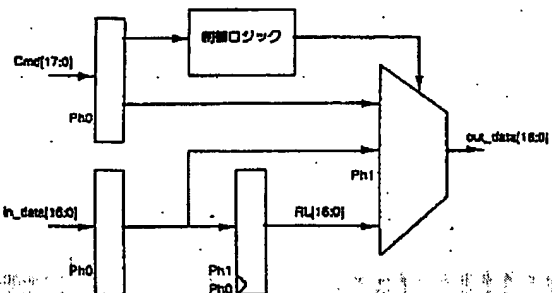
【图 143】



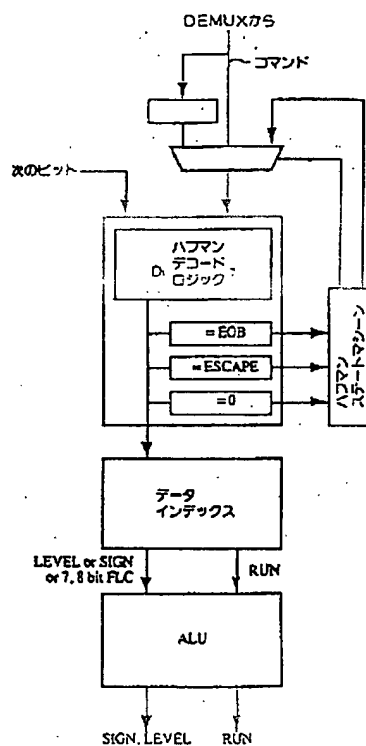
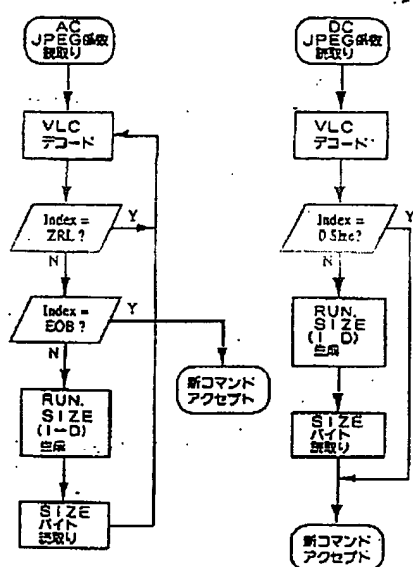
【图 129】



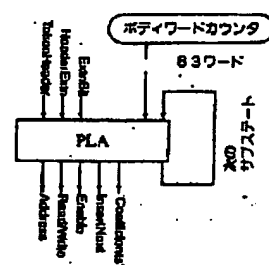
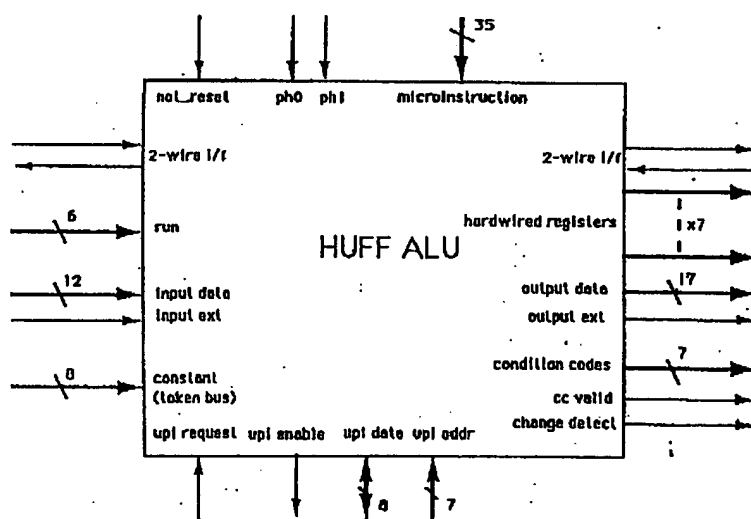
【图 1·3 1】



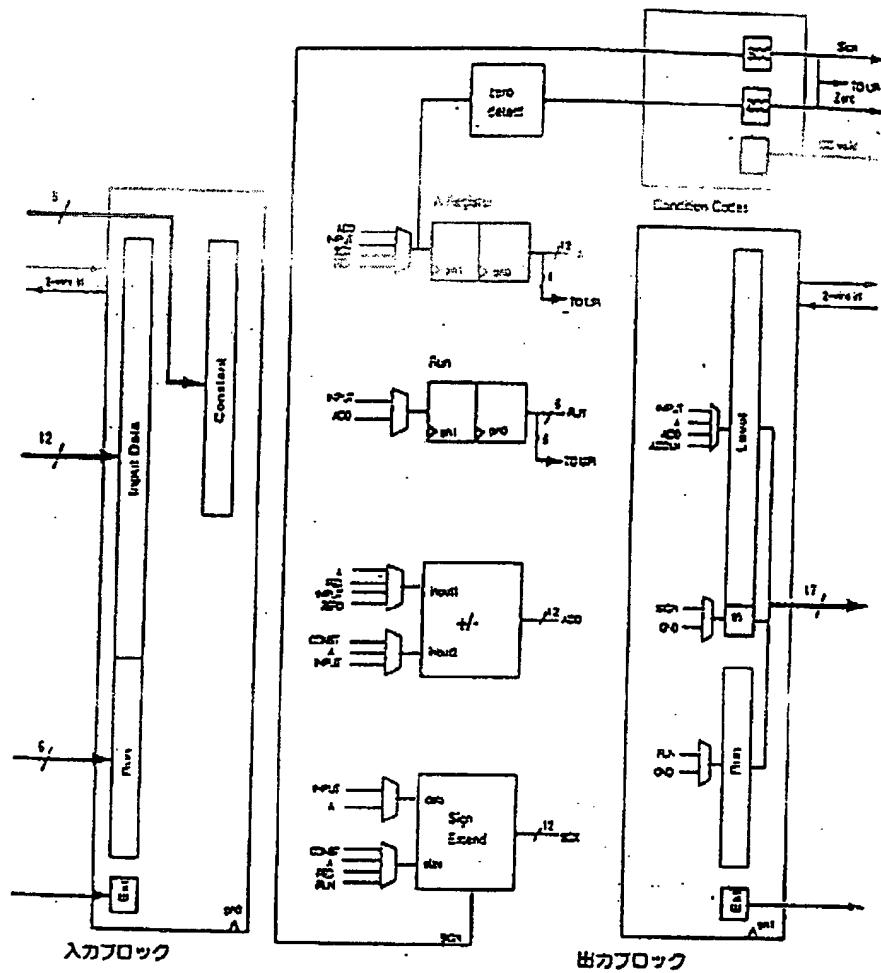
【图 133】



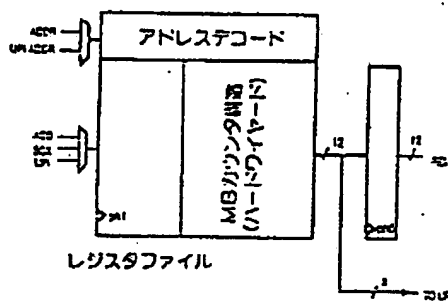
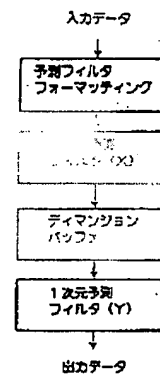
【图 144】



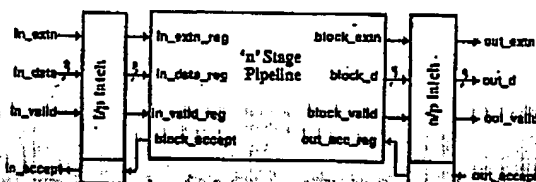
【図135】



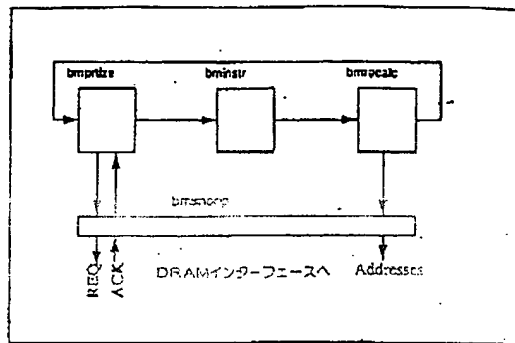
【図156】



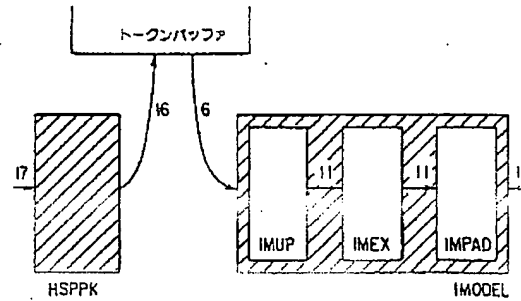
【図148】



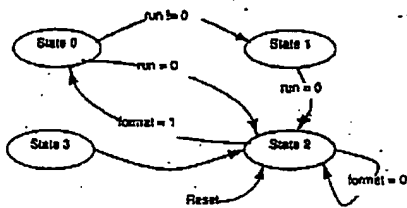
【図 1 3 6】



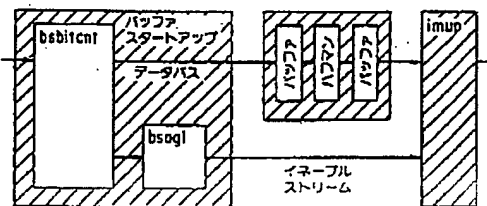
【図 1 3 7】



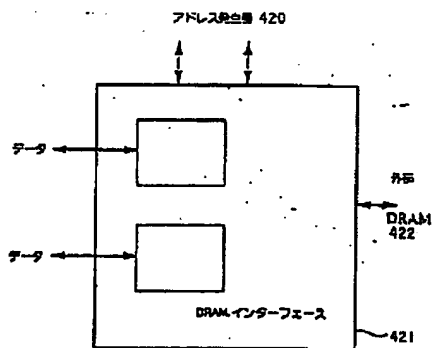
【図 1 3 8】



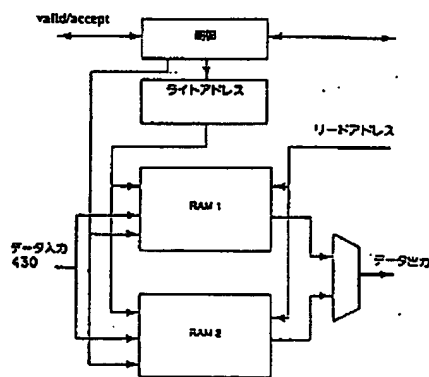
【図 1 3 9】



【図 1 4 0】



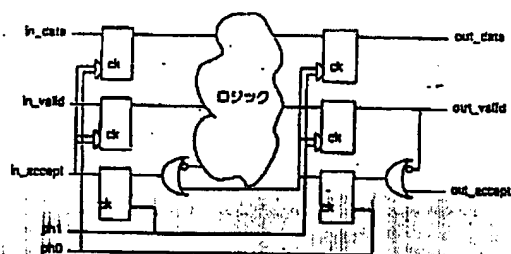
【図 1 4 1】



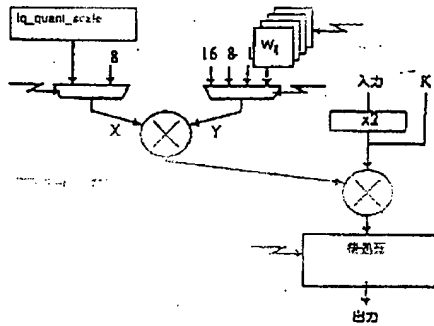
【図 1 5 8】

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

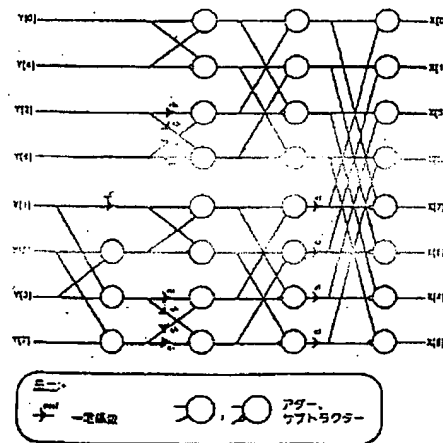
【図 1 5 2】



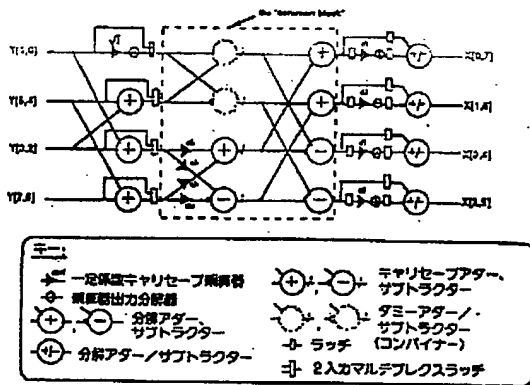
【図142】



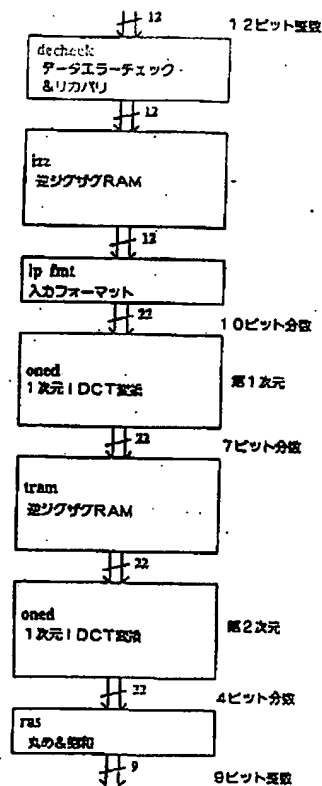
【図145】



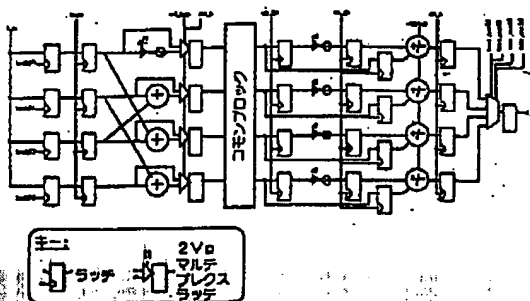
【図146】



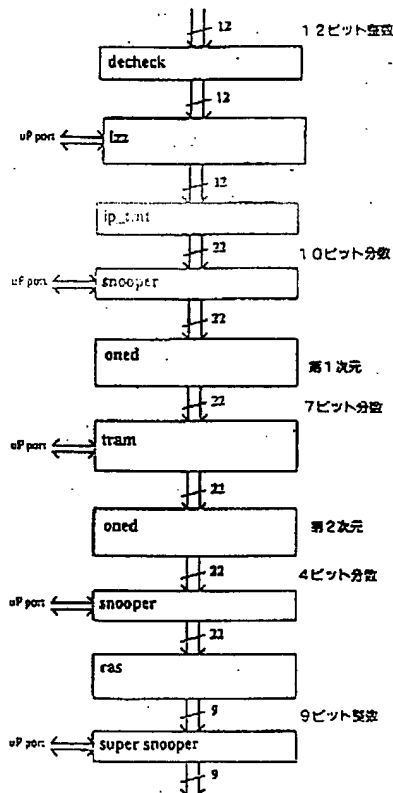
【図147】



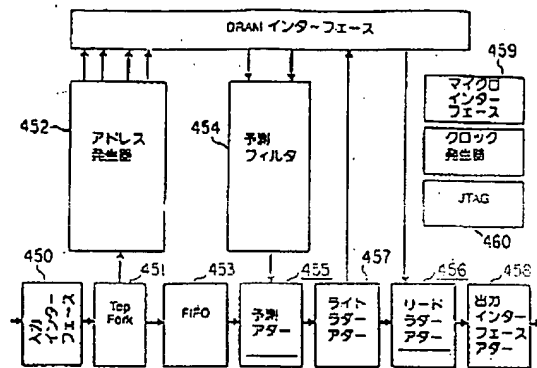
【図150】



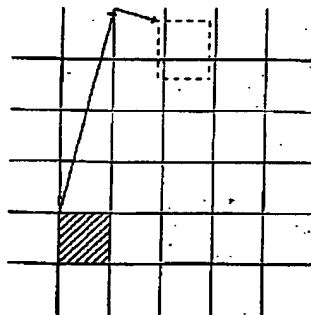
【図149】



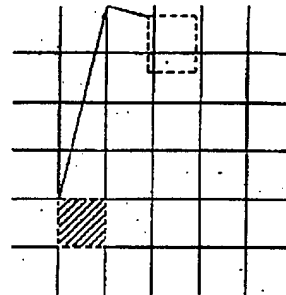
【図151】



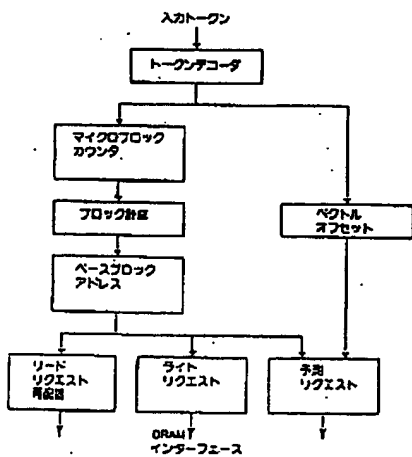
【図154】



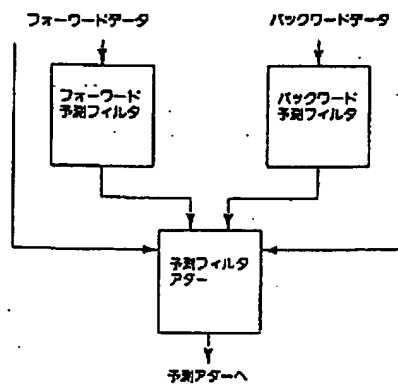
【図160】



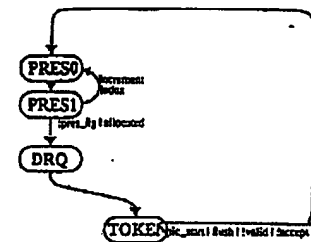
【図153】



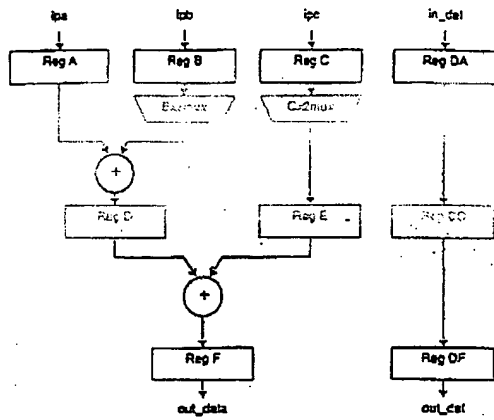
【図155】



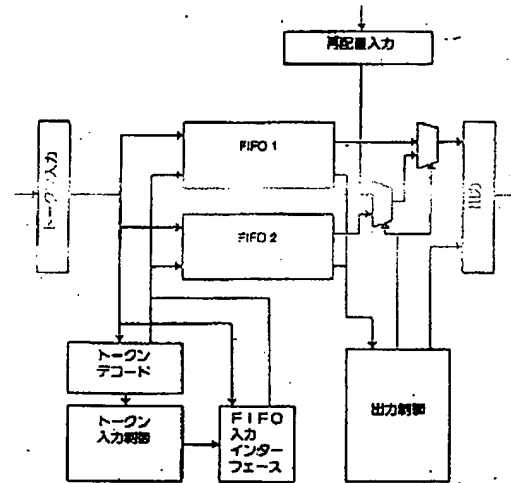
【図167】



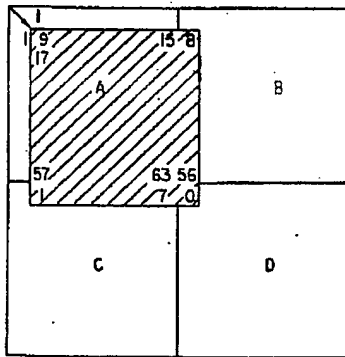
【図 157】



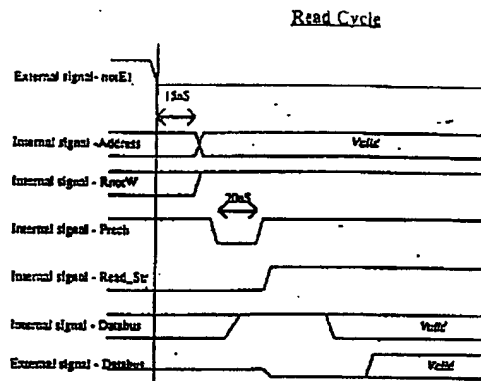
【図 159】



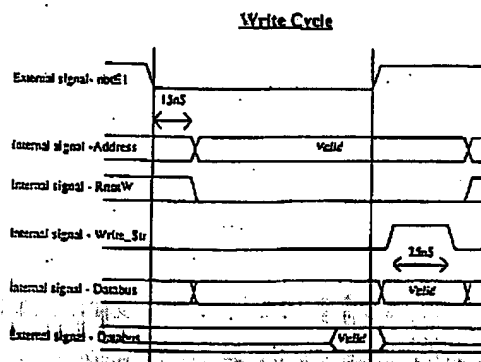
【図 161】



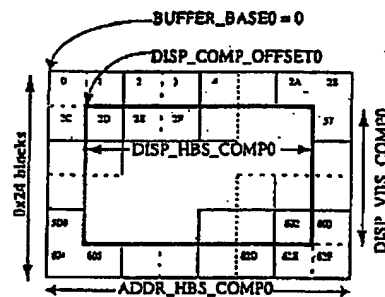
【図 162】



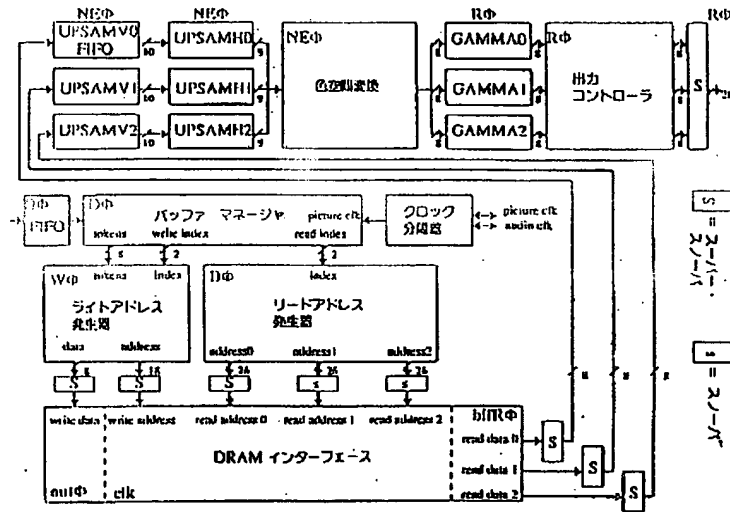
【図 163】



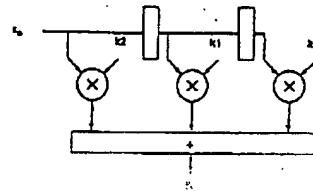
【図 169】



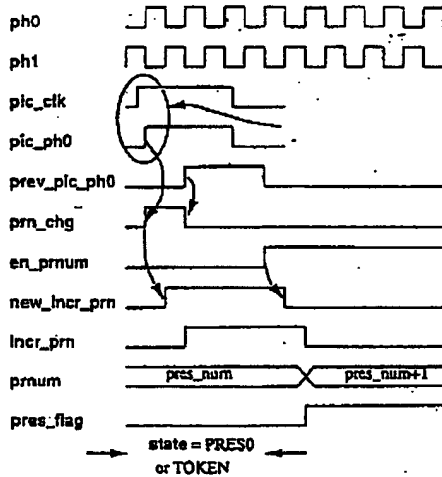
【図164】



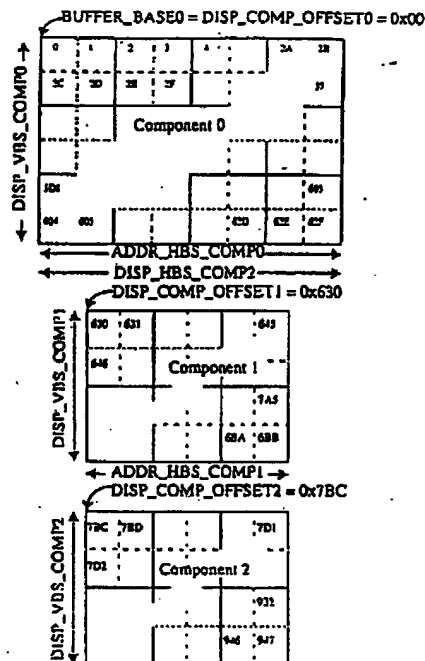
【図176】



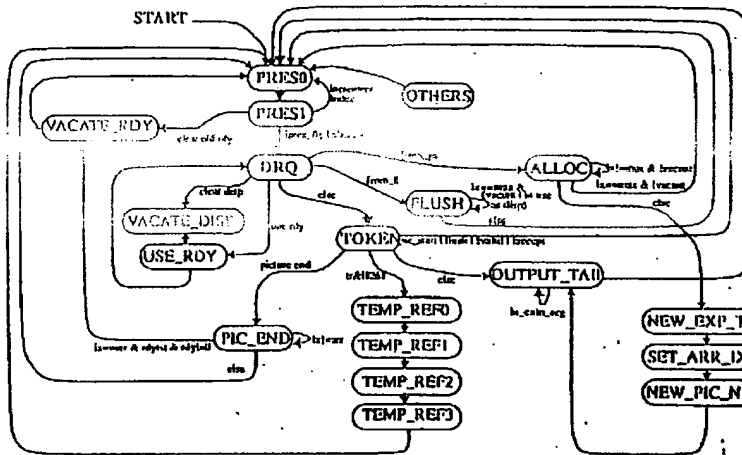
【図165】



【図168】



【図 166】



【図 170】

Buffer offset 0x00:

Component0 offset 0x000 +

00	01	02	03	04	05	06	07	08	09	0A	0B
0C	0D	0E	0F	10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F	20	21	22	23
24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B
3C	3D	3E	3F	40	41	42	43	44	45	46	47
48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B
6C	6D	6E	6F	70	71	72	73	74	75	76	77
78	79	7A	7B	7C	7D	7E	7F	80	81	82	83
84	85	86	87	88	89	8A	8B	8C	8D	8E	8F

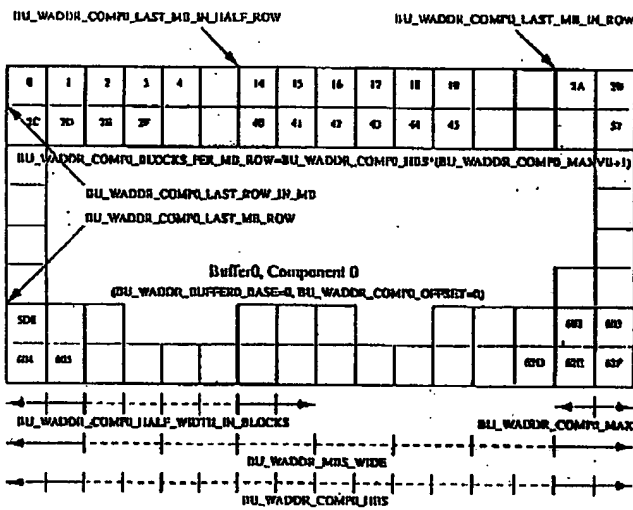
Component1 offset 0x100 +

00	01	02	03	04	05
06	07	08	09	0A	0B
0C	0D	0E	0F	10	11
12	13	14	15	16	17
18	19	1A	1B	1C	1D
1E	1F	20	21	22	23

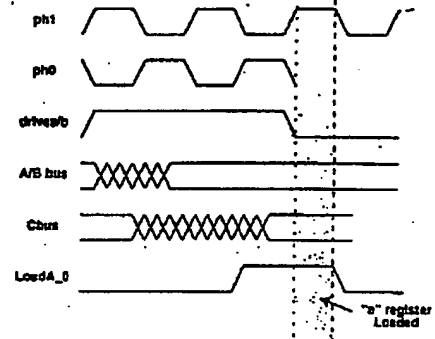
Component2 offset 0x200 +

00	01	02	03	04	05
06	07	08	09	0A	0B
0C	0D	0E	0F	10	11
12	13	14	15	16	17
18	19	1A	1B	1C	1D
1E	1F	20	21	22	23

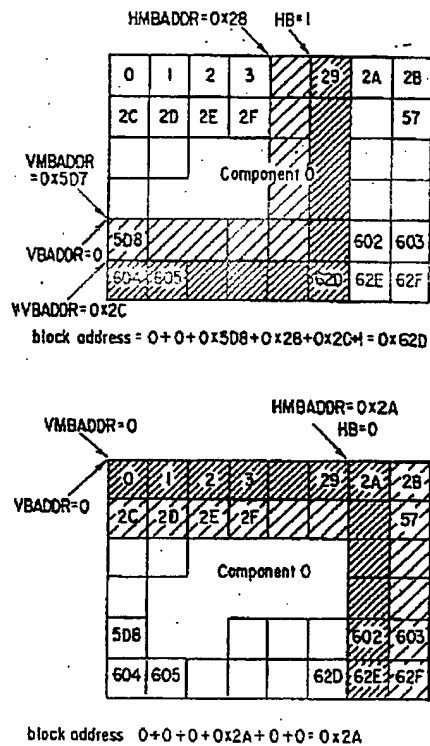
【図 171】



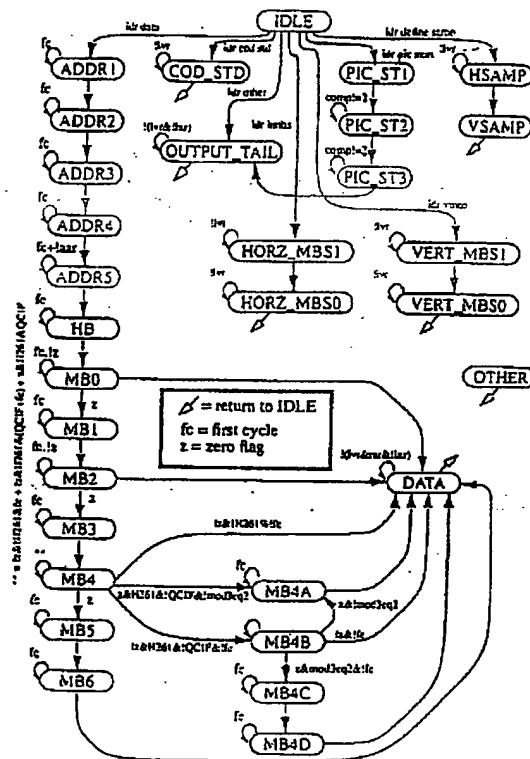
【図 175】



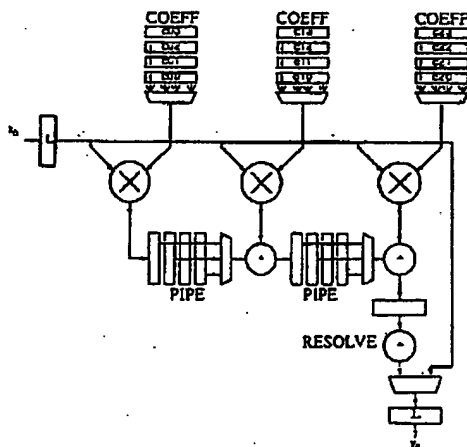
【図172】



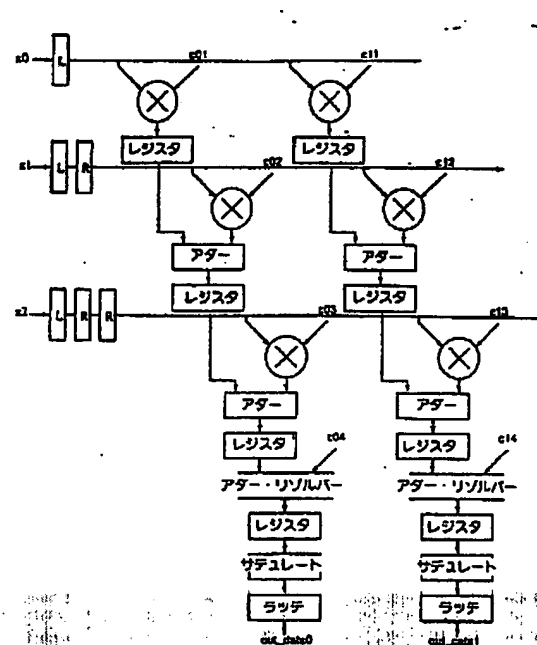
【図173】



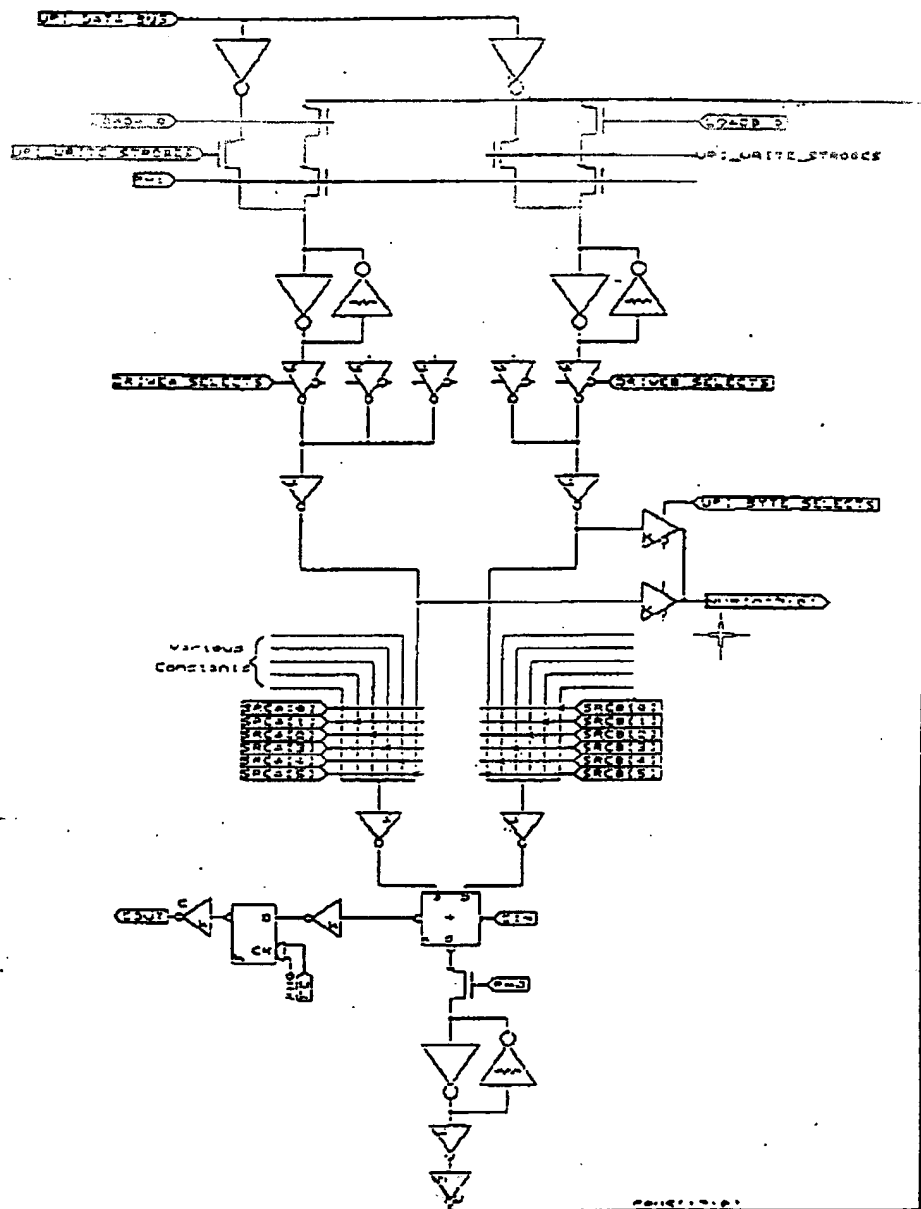
【図177】



【図178】



【図174】



フロントページの続き

(72)発明者 ウィリアム フィリップ ロビンズ
イギリス国、ジーエル11 5ビーイー、カ
ローセスターシャー、カム、スプリングヒ
ル 18

(72)発明者 アンソニー マーク ジョーンズ
イギリス国、ビーエス17 5ティーエフ、
ブリストル、エート、テンブラー ロード
81

(72)発明者 アンソニー ピーター ジョーン クレイ
ドン
イギリス国、ビーエー2 6 ビーゼット、
エイボン、バス、シドニー ビルディング
ス 14

(72)発明者 マルティン ウィリアム ソザラン
イギリス国、ジーエル11 6 ビーディー、
グローセスターシャー、ダズレイ、ステ
インチコーム、ウィク レーン、ザ ライ
ディングス (番地なし)

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☒ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.